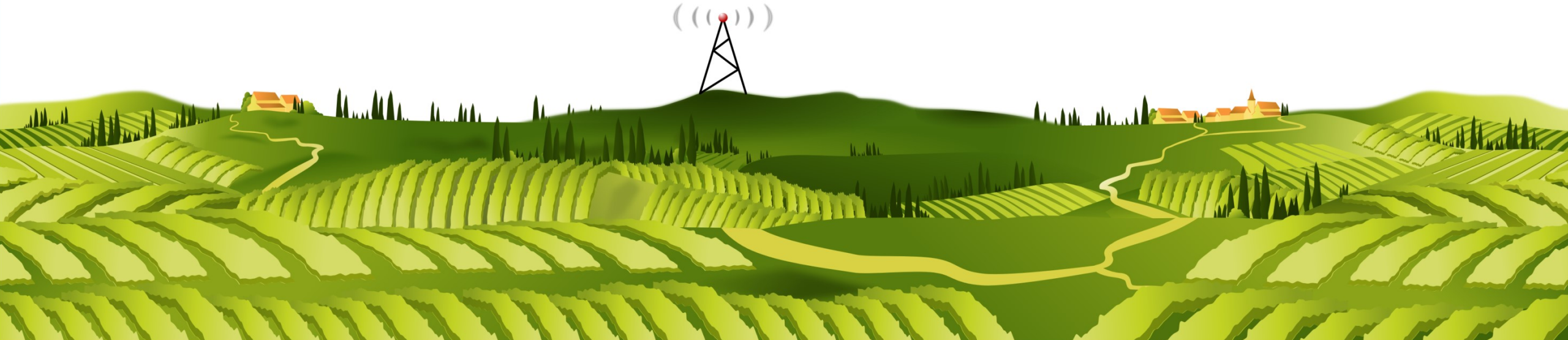
THE ULTRASWITCH

- a remoteqth project by DM5XX -



remoteQTH.com

- OPEN-SOURCE Hardware + OPEN-SOURCE Firmware
- OK1HRA, Dan | OK2ZAW, Jan | OK1CDJ, Ondra | DM5XX, Mike
- https://remoteqth.com + qro.cz + hamshop.cz = OL7M
- K9AY
- Beverages: Top band, single band, bi-directional 2-wire, 4-way bi-directional,…
- Switching: 6To2/6To1/4To1, 6To2+Triplexer, 1To2/1To3 Stackmatch, 12x4,…
- Variable ATT, SO2R, M/M, Controller (on-site/remote), Custom Contest Hardware…

DM5XX

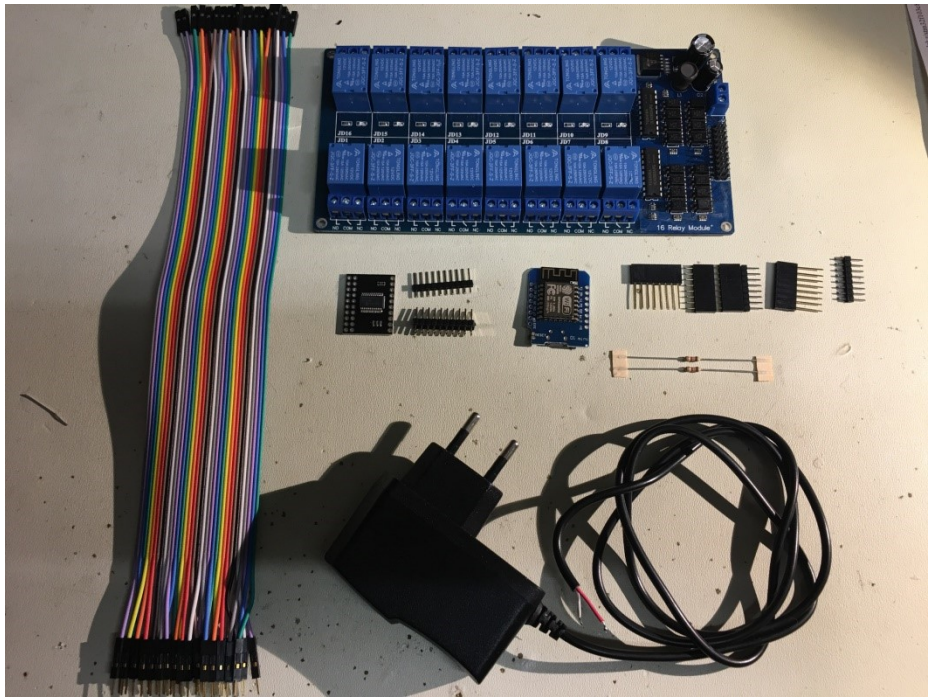
- Radio amateur since 1990 (OK8XX, DM5X, ex. DG3NED, ex. DL3NED) living in Nuremberg
- Senior Software Engineer
- IT-Project Lead @ ADITUS GmbH
- Since 2015 Lead Firmware Developer @remoteqth.com
- Main responsibility for embedded microcontroller firmware

The Ultraswitch

- Wifi based & inspired by mega-webswitch (63-CH remote-switch)
- Part of the 256-Port-Inband-Antennaproject @ OL7M
- Simple hardware direct from China (see ebay.com :P) – no custom PCBs!
- ESP8266-12F („Wemos D1 mini“)
- Modular: Up to 8 Boards
- Web-UI & Web-Config
- RESTful and simple extendable thanks to ist web based architecture
- Pre-Programmed kits available from dm5xx@gmx.de

Needed Hardware

- 16-CH 12V-Relayboard
- MCP23017SMD on a PCB (I2C=>16 Port)
- Wemos D1 mini (ESP8266-12F, Controller)
- 2x 1.8K/2.2K
- DuPont connector
- 12V-Power supply
- Computer+MicroUsb-Cable

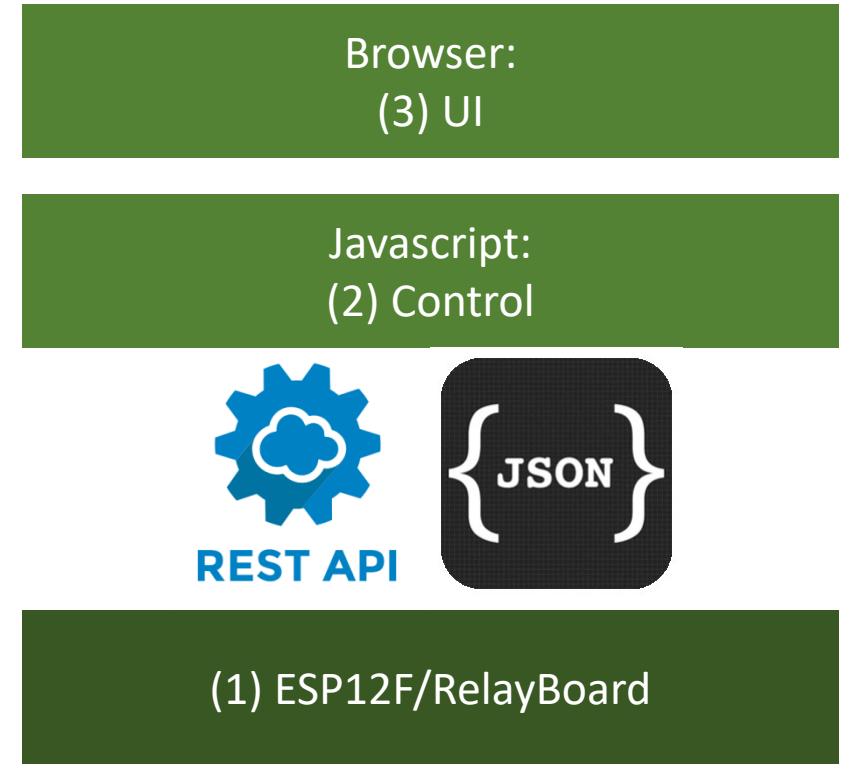


The architecture

Separation of Concerns:

- Hardware layer (1)
- Hardware based functionality (JSON) (1)
- Function layer (JS) (2)
- Presentation layer (UI) (3)

=> aka 3-Layer-Model



Pros and Cons

Pro:

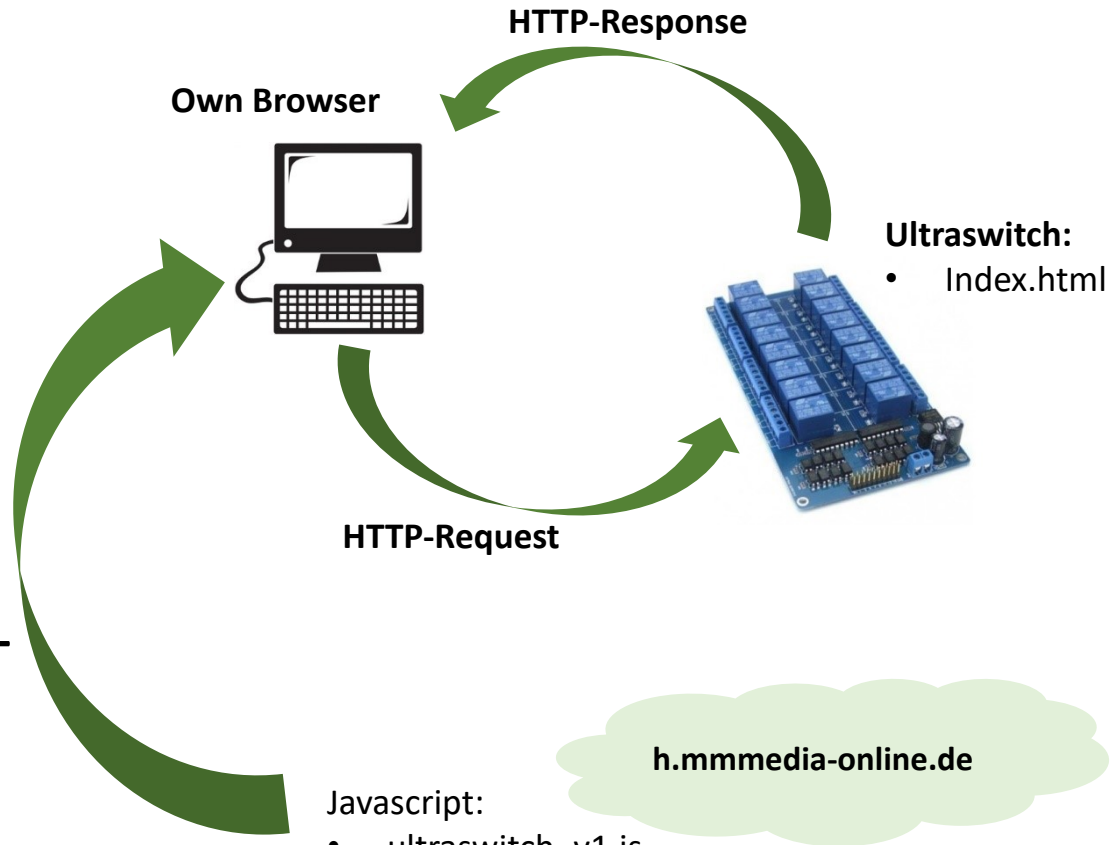
- Simple extendable, not dependend on the UI
  (Web-App, Windows-App, Mobile-App, Linux-App,…)
- Seperation of hardware, logic and (business) functions
- Centralized, modular, web technology, ready for future demands
- Add your own functions and/or UI (Customizing)

Cons:

- Security concerns are up to you (use a VPN if using remote access)
- The more you are customizing, the more you need to know (learning curve)

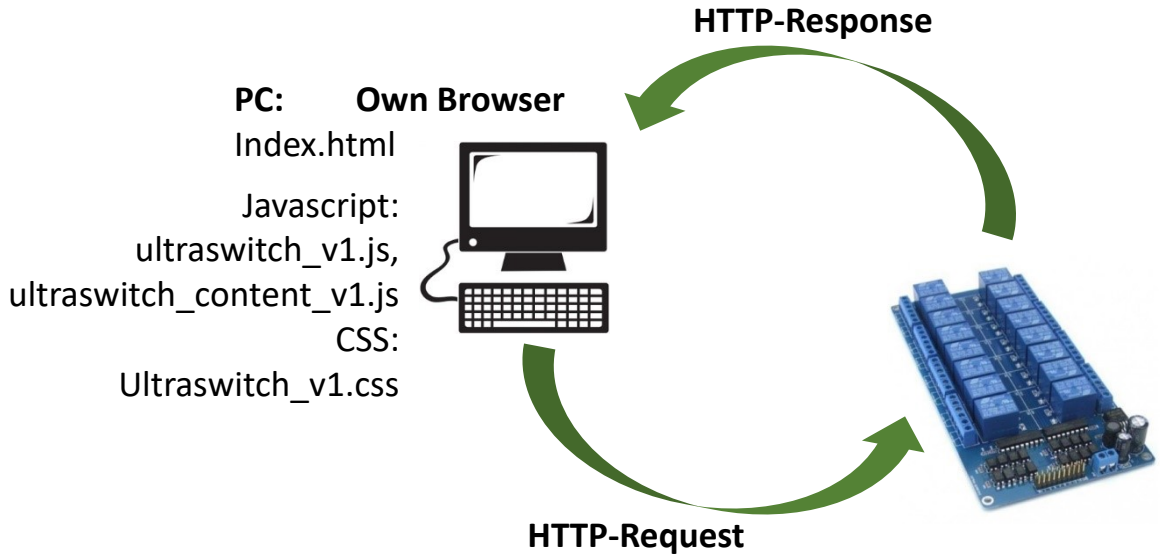
Usecase 1 – „Out of the box“

- No changes to be made
- All already available resources („h.mmmedia-online.de“) are used.
- Just the configuration of the switch is necessary (IP-Adress)
- Software is always up-to-Date
- Cons: You cannot change something – including the labels.



Usecase 2 – Using ur own „Desktop-Application“

- Easy to custimize
- All resources are local on ur PC
- Everything is customizable
- No own webserver is needed
- Cons: No updates out of the box, no longer multi-user capable (Logic/Infos are on ur own pc). Not mobile accessable.

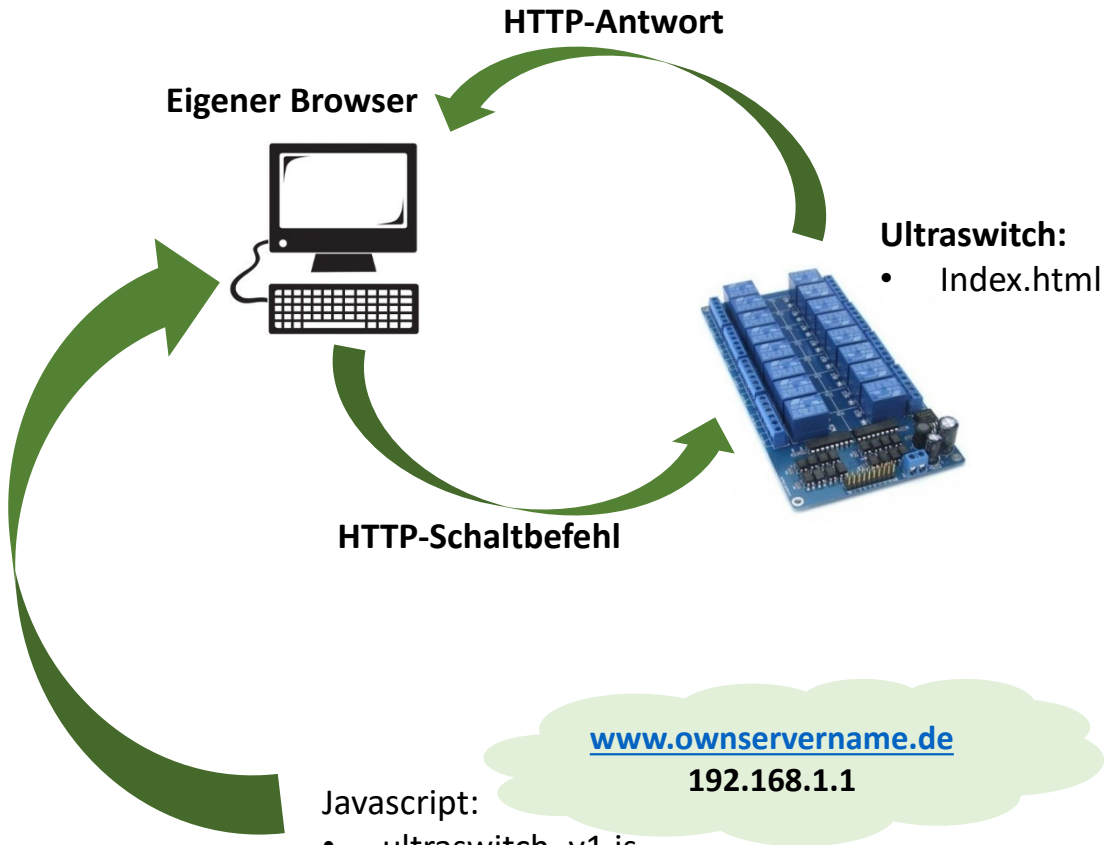


Usecase 3 – „Own Webserver“

- All advantages of Case 1 and 2
- Multi user capability
- Server could be in your own network

Cons:

- Webspace needed
- Knowledge in web techniques needed
- Updates no longer out of the box

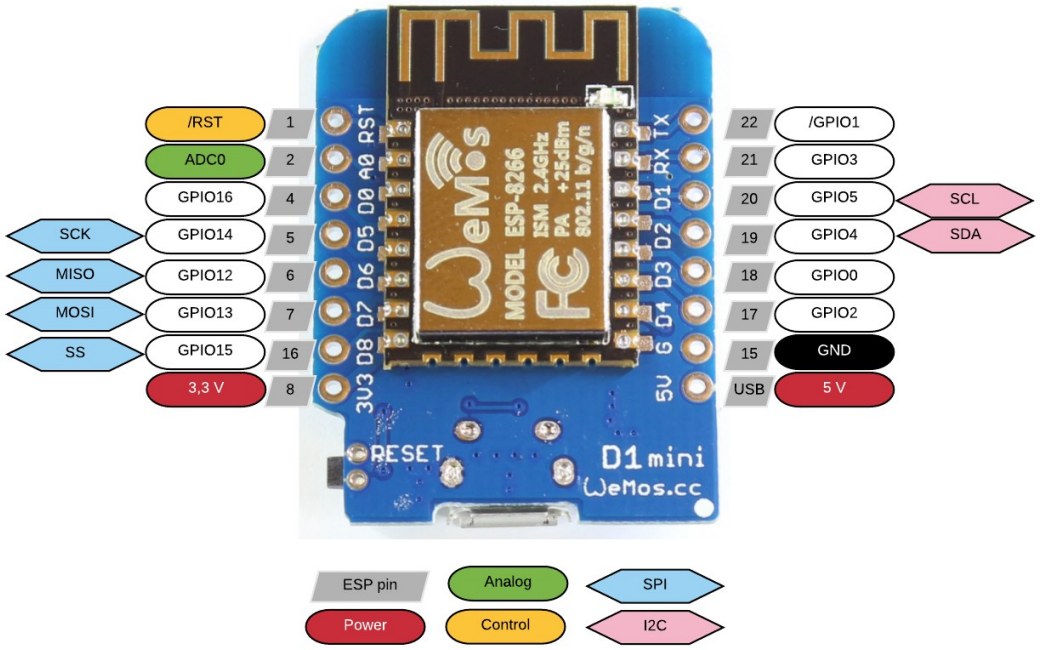


Good to know

- Computer networks
- Network protocols
- JSON
- HTML, CSS, Javascript
- Introduction into REST

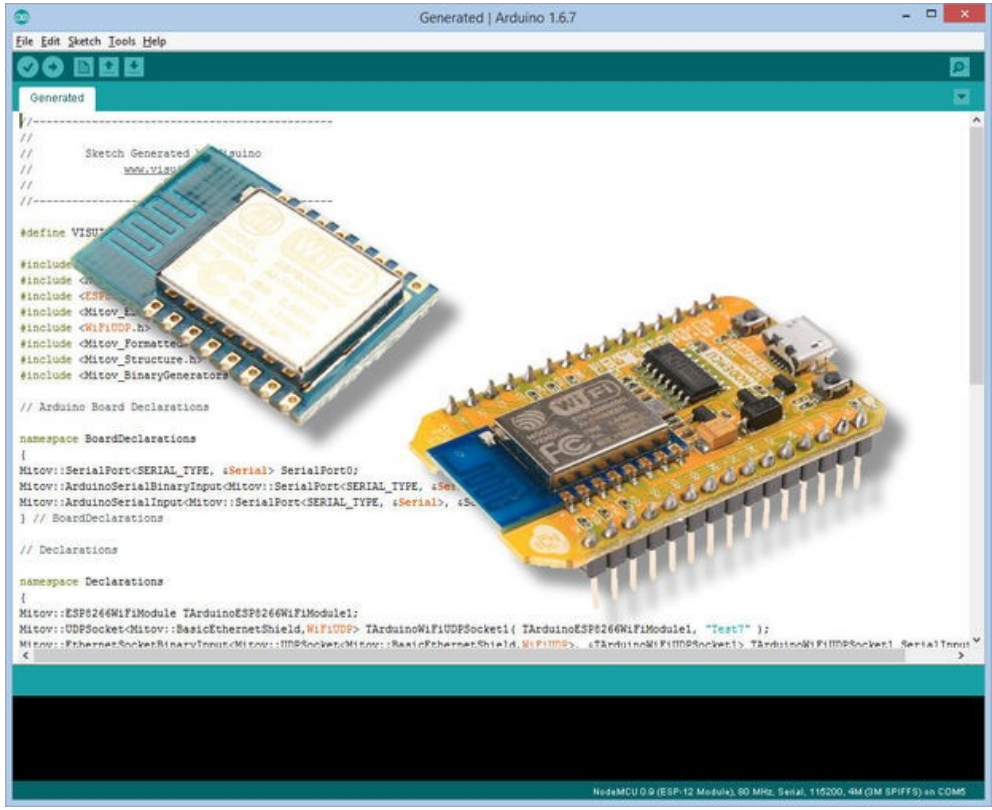
Hardwareplattform – ESP8266-12F

- 802.11 b/g/n, Wi-Fi 2.4 GHz, support WPA/WPA2
- Integrated low power 32-bit MCU @80mhz,4MB Ram, 92kb
- Integrated 10bit ADC
- Integrated TCP/IP protocol stack
- Integrated TR switch, balun, LNA,
- power amplifier and matching network
- Integrated PLL, regulators, and power management units
- Supports antenna diversity
- Support STA/AP/STA+AP operation modes
- Support Smart Link Function for both Android and iOS devices
- SDIO 2.0, (H) SPI, UART, I2C, I2S, IRDA, PWM, GPIO, STBC, 1x1 MIMO, 2x1 MIMO
- Deep sleep power <10uA, Power down leakage current < 5uA
- Standby power consumption of < 1.0mW
- [Datasheet]



Arduino-IDE

- Installing Arduino IDE
- https://learn.sparkfun.com/tutorials/esp8266-thing-hookup-guide/installing-the-esp8266-arduino-addon
- http://arduino.esp8266.com/versions/2.3.0/package_esp8266com_index.json
  (Use 2.3.0! 2.4.0 has a bug)



Libraries

- Installing Additional Arduino Libraries (content of needed_libraries is copied to your local arduino library folder)

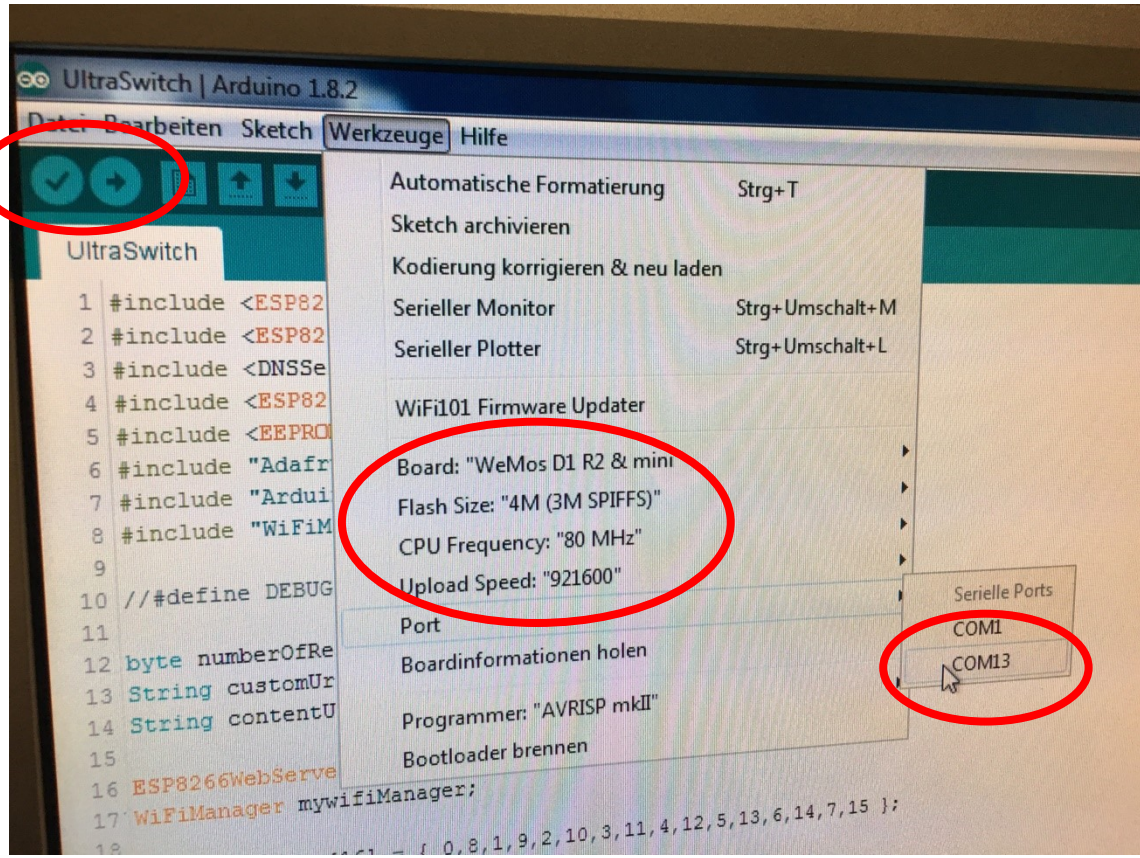
After IDE-Installation look for your project files…

- C:\Users\mmmedia\Documents\Arduino
  => copy the folder „Ultraswitch“ into this folder
- C:\Users\mmmedia\Documents\Arduino\libraries
  => Content of “Ultraswitch/needed_libraries“ are copied to this location
- Copy to your desktop: (for later use)
  => Folder “ultraSwitch_web”

Lets go: 8266-12F

- Board ESP added?
- Libraries copied?
- Sketch loaded? (ultraswitch.ino)
- Preferences adjusted?
- Serial port connected?
- 12F connected to usb?

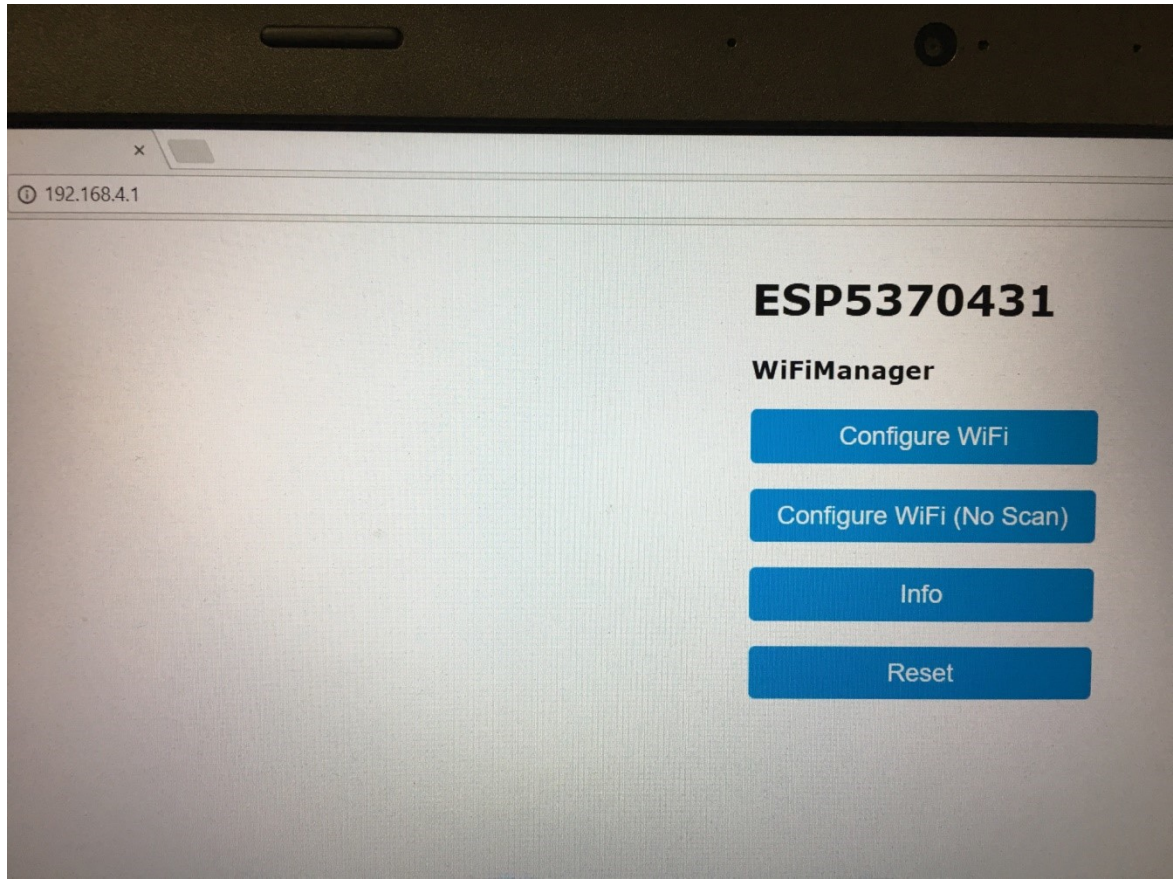
Yes? => Verify > Upload!



engl.: „Tools“

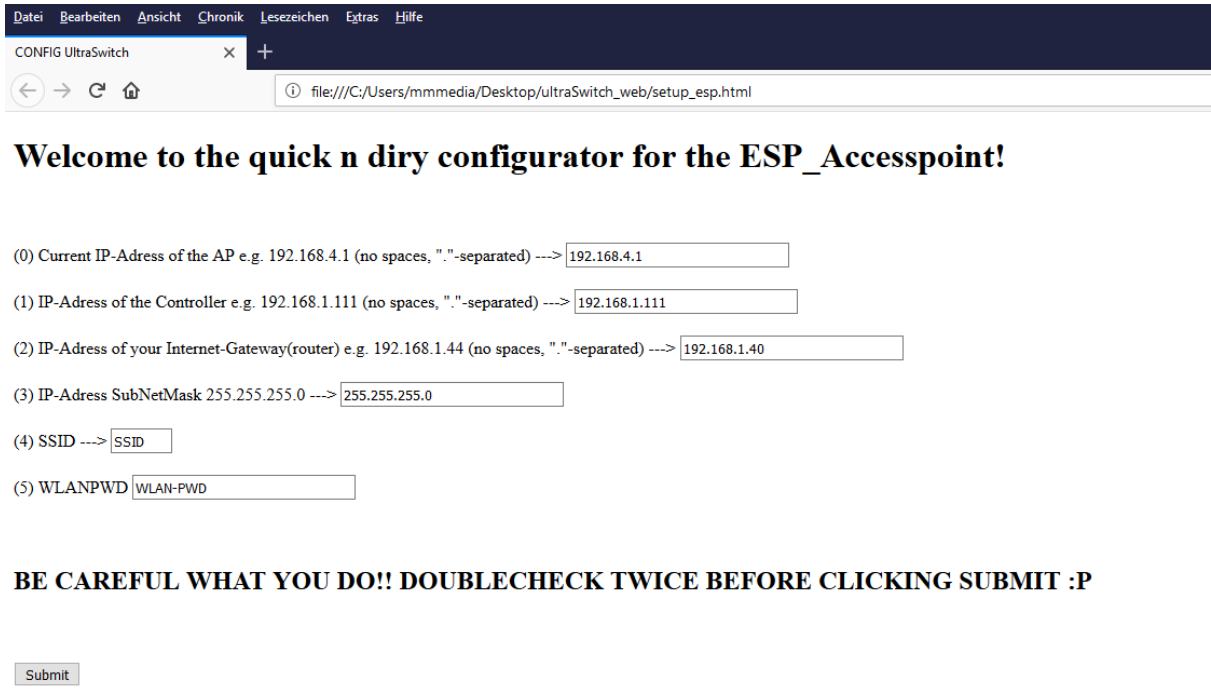
Connecting to the 12F – AP-Mode 1

- Search for the Wifi-Network ESP<number> (with your smartphone)
- **Connect to this network** ESP<number>
- Open your Browser (Default: 192.168.4.1) – do nothing more yet!



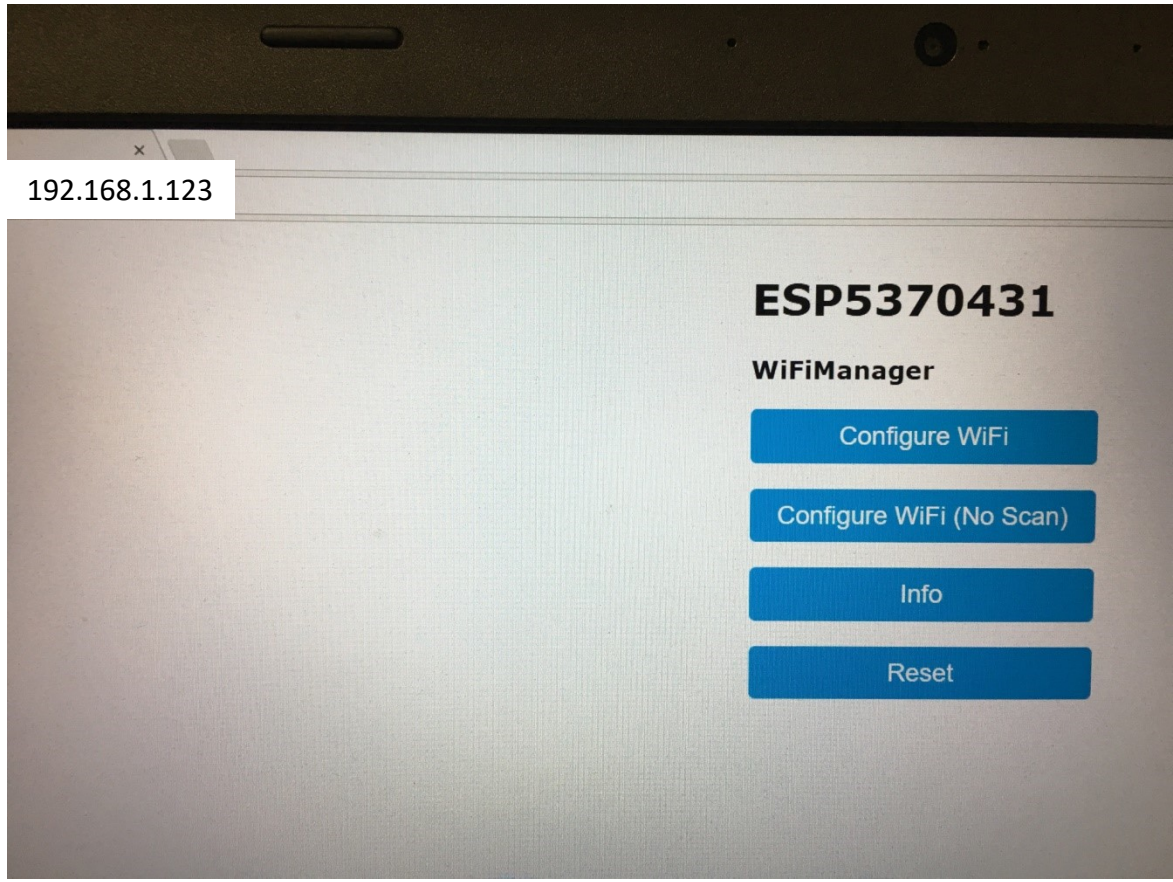
Connecting to the 12F – AP-Mode 2

- Execute setup_esp.html in the folder „Ultraswitch_Web“ with a double click.
- Change ur data – give everything its needed ip, mask, ssid, wifi-pwd
- Submit!



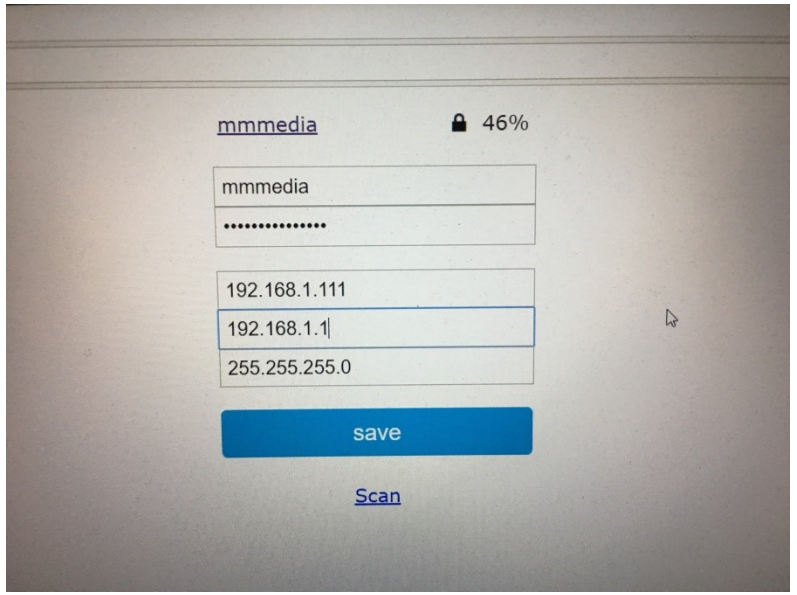
Connecting to the 12F – AP-Mode 3

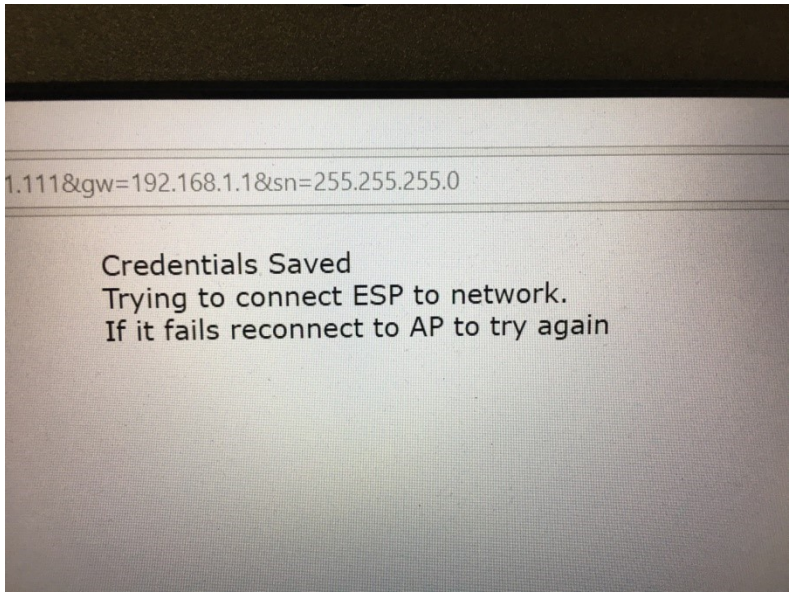
- **Connect your normal wifi**
- Open your 12F-Ip in your browser (z.B. 192.168.1.123)
- „Configure Wifi“



Connecting to the 12F – AP-Mode 4

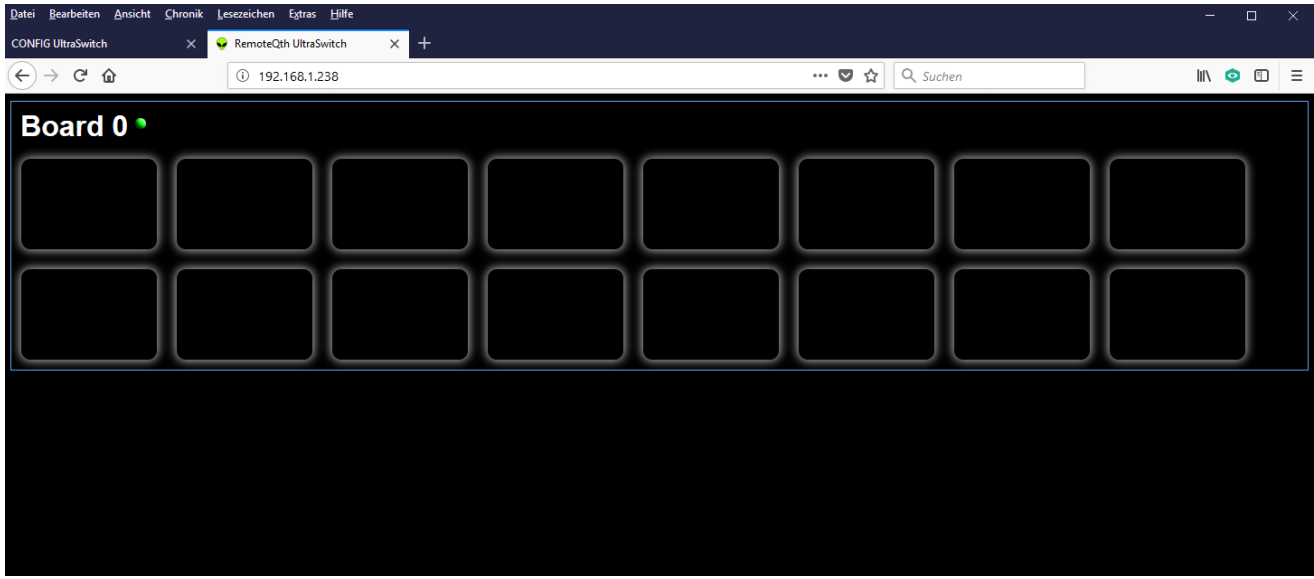
- „Configure Wifi“:
  Give SSID and Wifi-PWD again!
- Double check your data! Save.





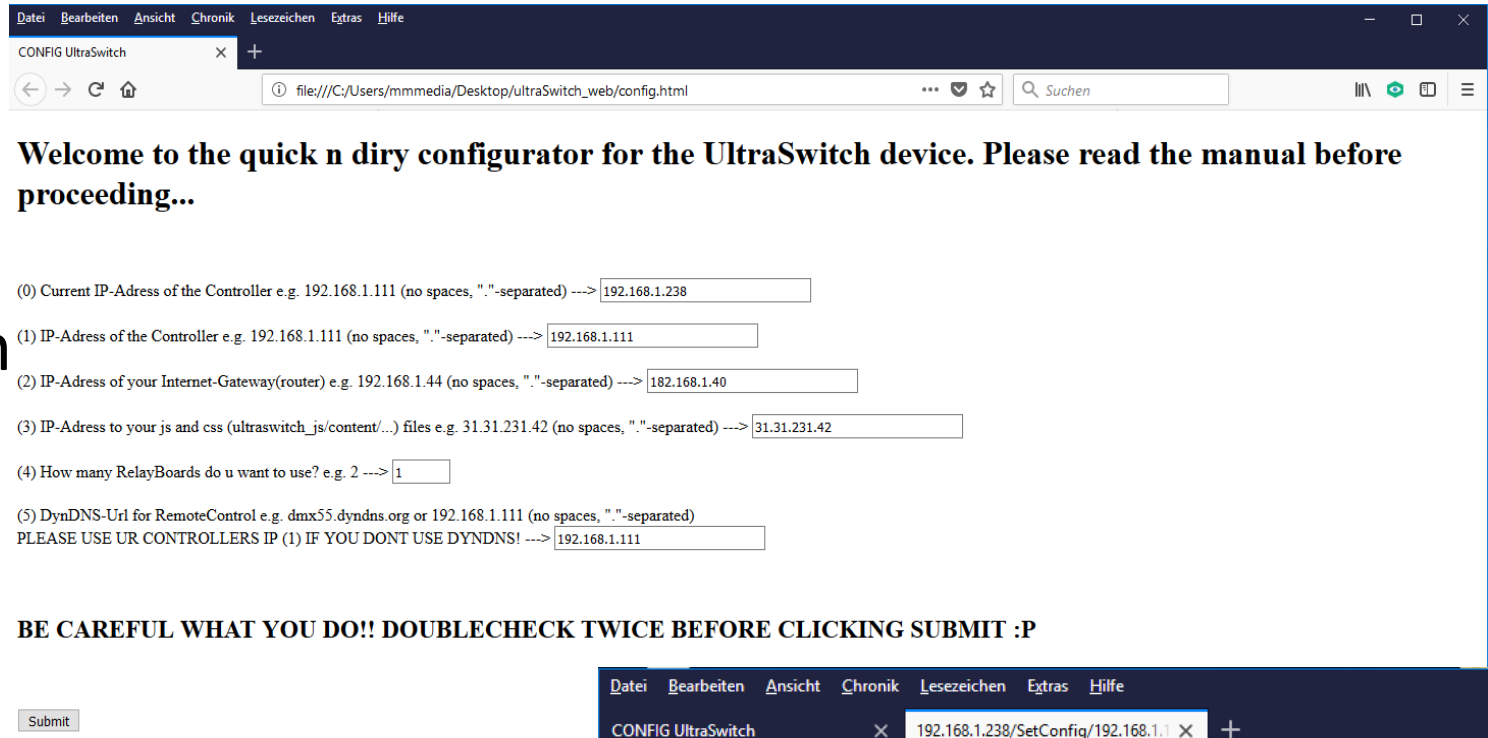
Connecting to the 12F – First contact

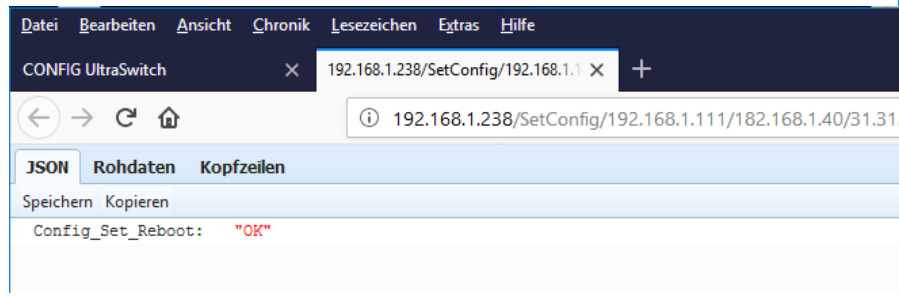
- Call your Ultraswitch by its given IP.
- Troubleshooting: Ultraswitch not found?
  Advanced IP-Scanner helps!
  Scan your network to find ist ip an try to set it again!
- But you can see no buttons, …



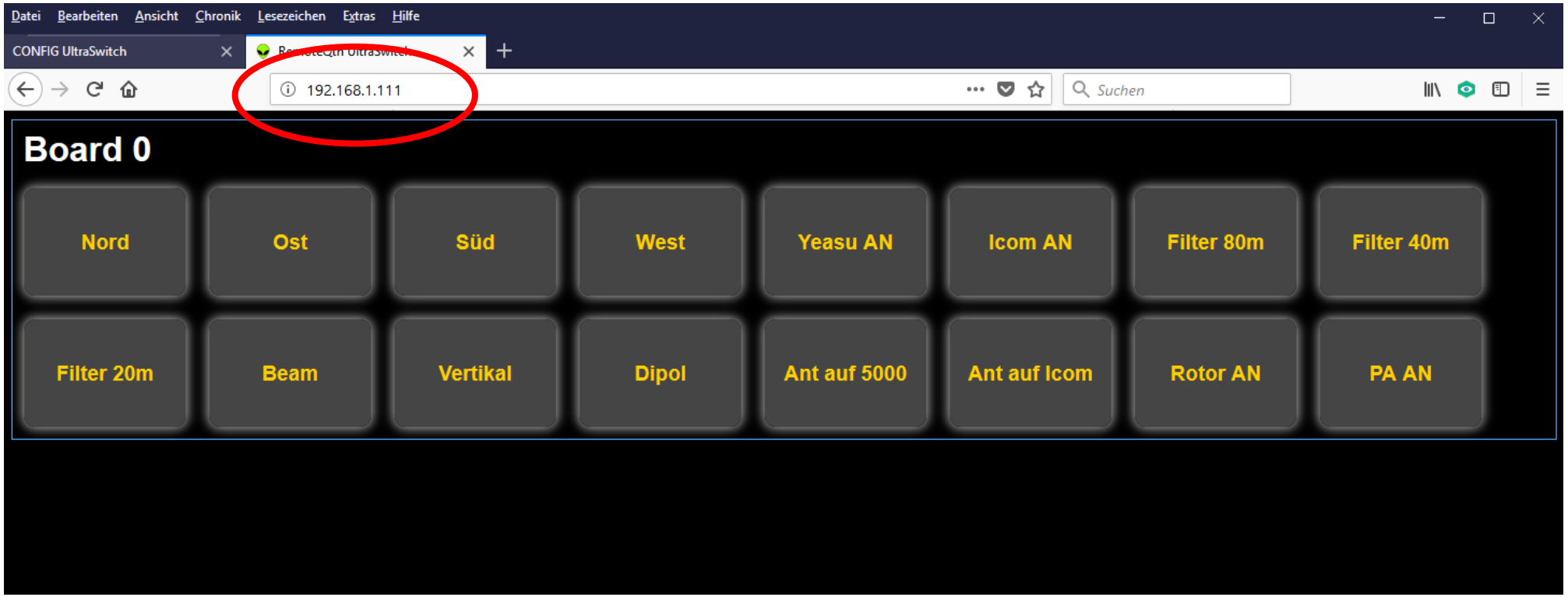
Connecting to the 12F – IP-Settings Servermode

- Call Config.html
- You need to know the IP of your Ultraswitch now!
- You can give the ultraswitch another ip if u want
- Add all needed info
- Submit!





Final IP-Adresse of your Ultraswitch: The UI is working!



Something is wrong in your sketch... DEBUG!

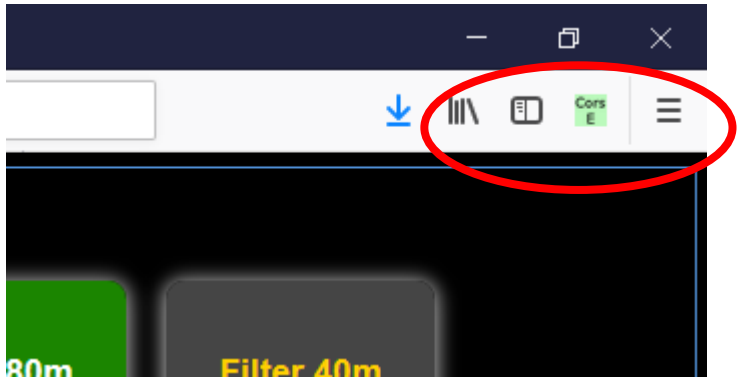
- Uncomment ultraswitch.ino:
  //#define DEBUG to #define DEBUG (remove „//“)
- Upload your sketch
- Tools => Serial monitor
- Adjust Monitor Serial Speed from 9600 to 115200
- Reset your Wemos (little switch on your Board)
- Serieller Monitor shows your ip and switched pins...
- Comment //#define DEBUG after you are done (upload again)

CORS – Cross Origin Request

- Due to a Ajax-Request i could be, that your browser is blocking those cross origin requests, if you start the index.html from your desktop (file://file:///C:/Users/mmmedia/../../index.html)

- Use: CORS Everywhere (Firefox)
  => Add to your Firefox
  => Activate CORS E („CorsE“ icon will be green)

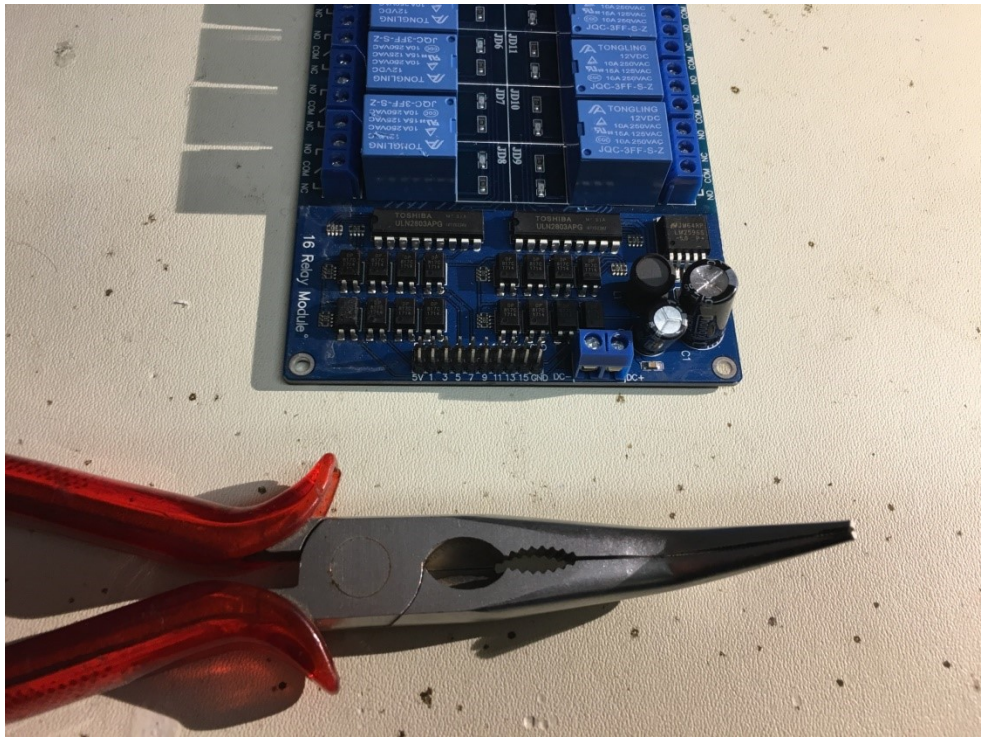
- Use Firefox!



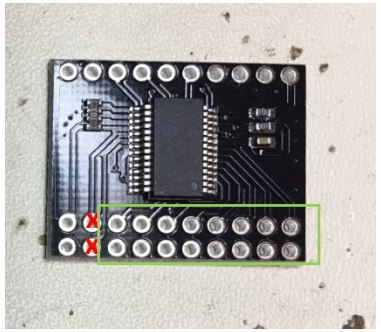
Wiring up your Relaysboard… 1

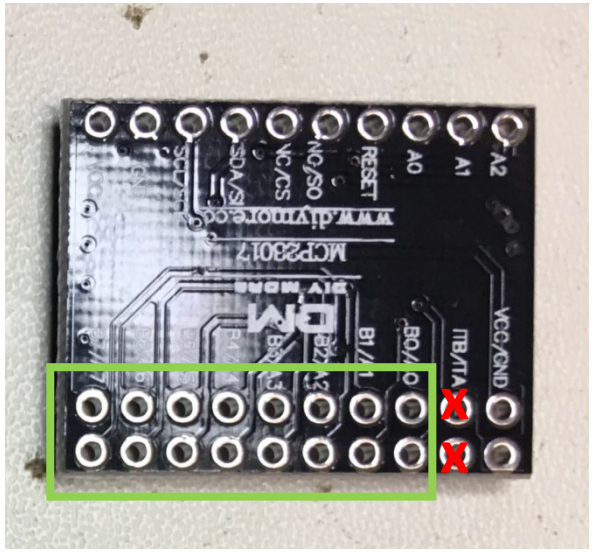
Use your pliers to bent your header pins of your relay board so they fit into the MCP-Board

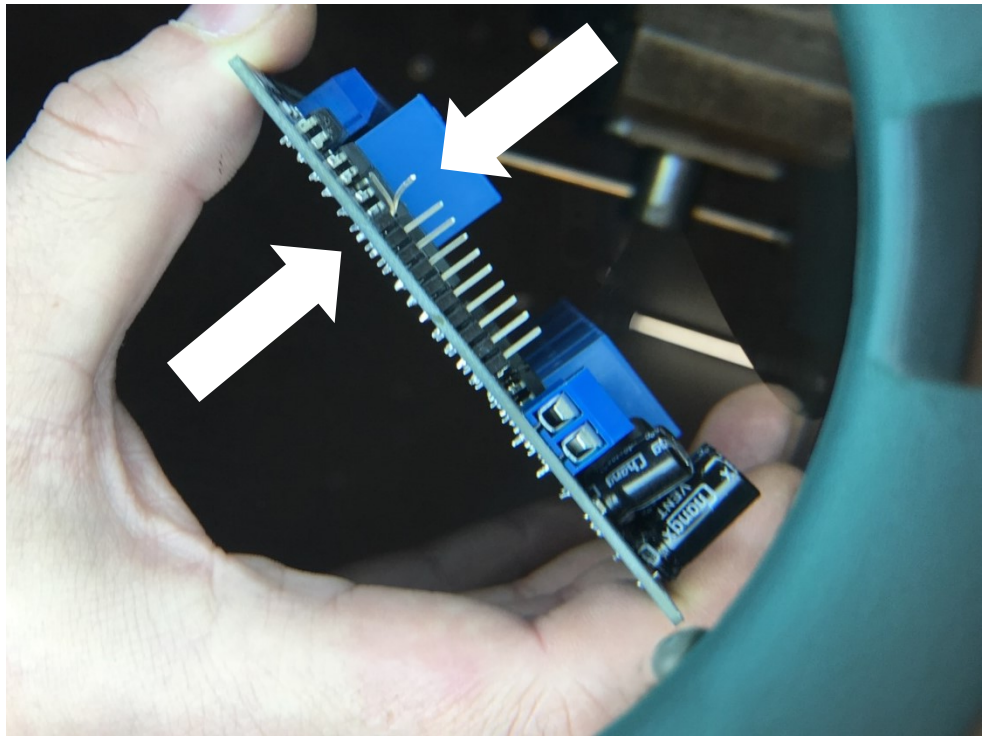
Why? 2 Pins (3,4) from your MCP-Board need to be unconnected! Its not so easy..



Wiring up your Relaysboard… 2

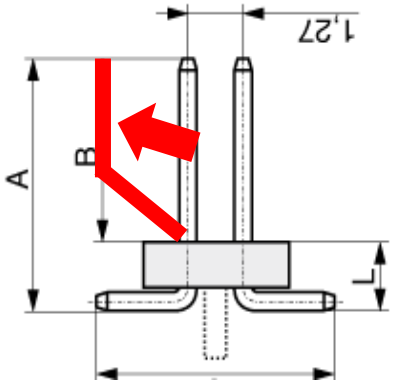




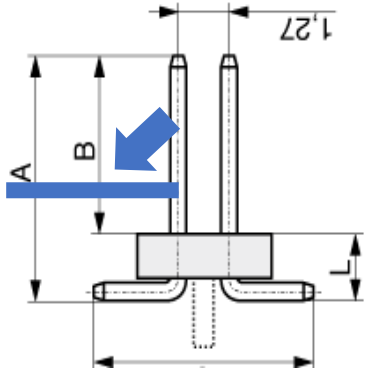


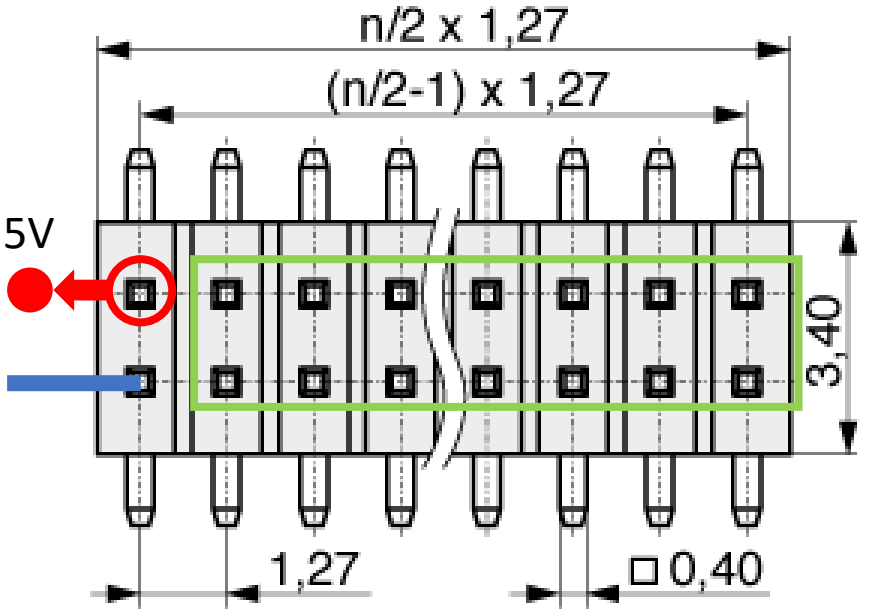
Wiring up your Relaysboard… 3

- Pin 1: „Bent“ one grid to the left!



- Pin 2:
  Bent completely down!



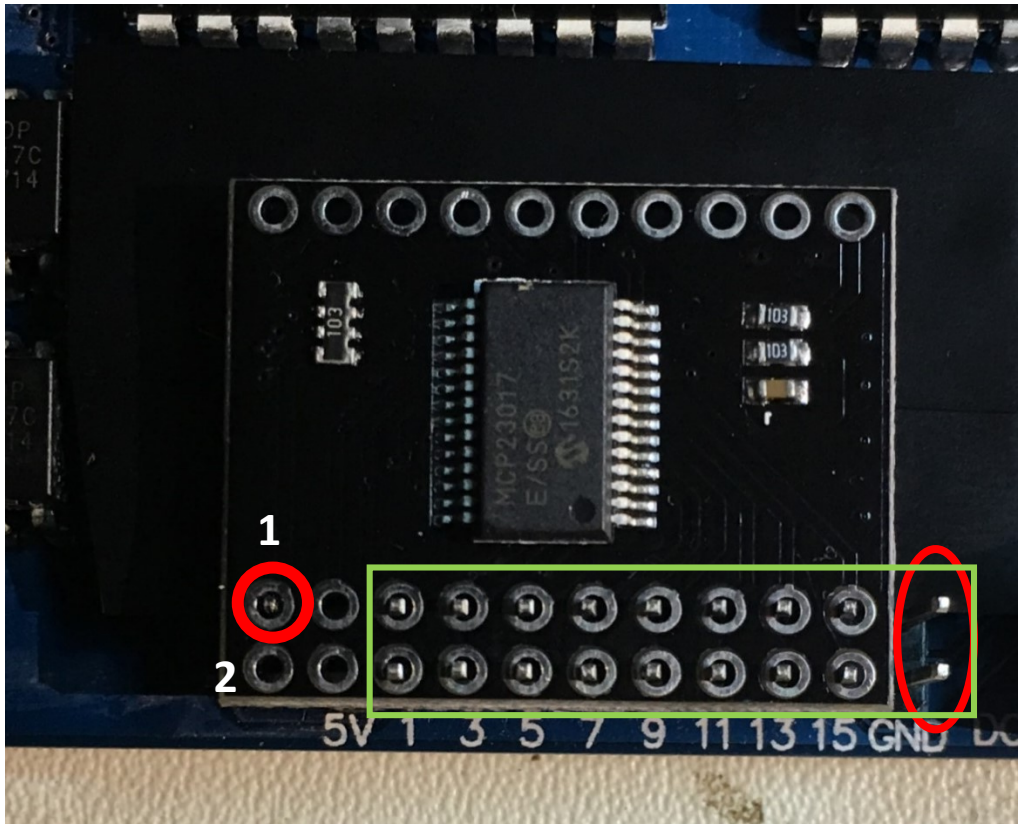


Wiring up your Relaysboard… 4

Connect the MCP-Board:

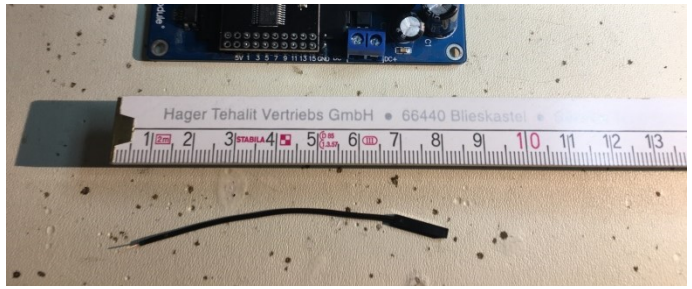
- -1- the „S“ bent pin is visible. To be soldered later!
- -2- complete bent pin is not visible
- MCP Pin position ¾ is not connected.
- On the right: Free GND-Pins (see: later))

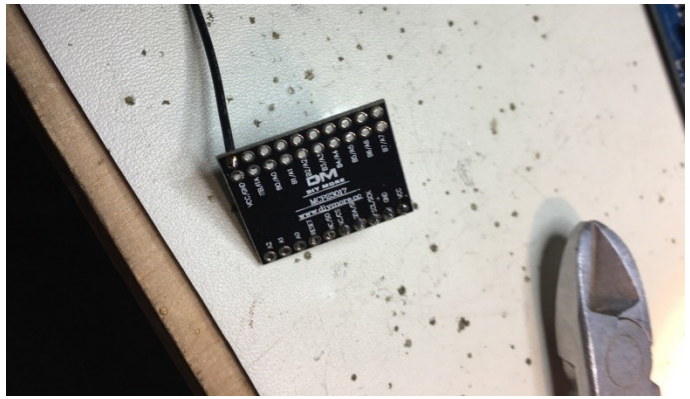
Important: Do not solder anything yet!
Just see if it fits already. Maybe re-bent.

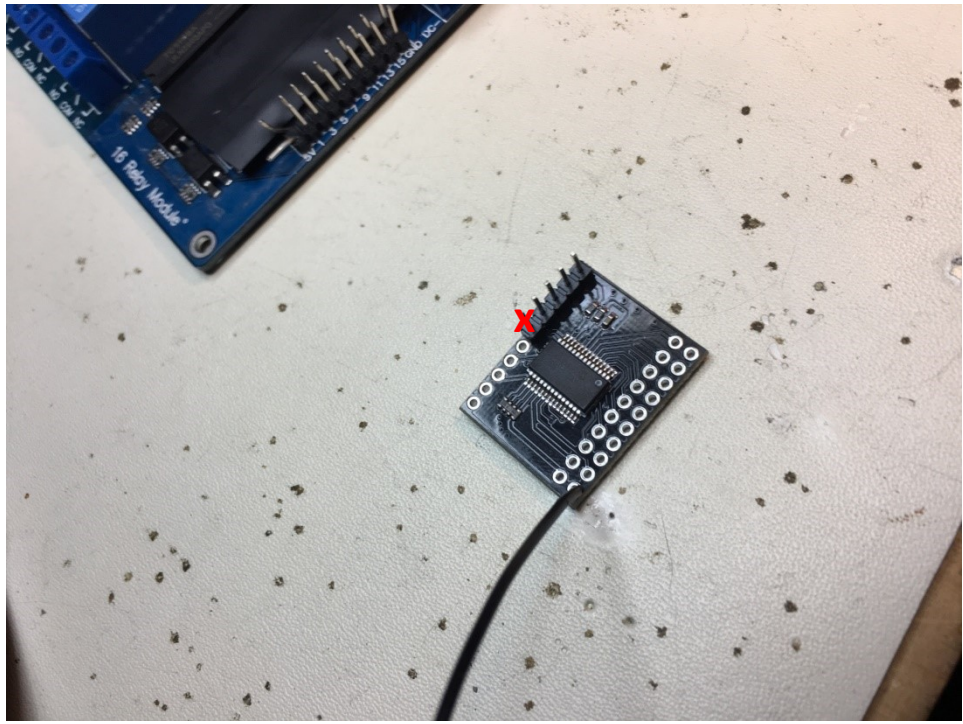


Connecting the MCP-23017 board

Use 4 Pin-Header and 1x GND-Wire and solder them:

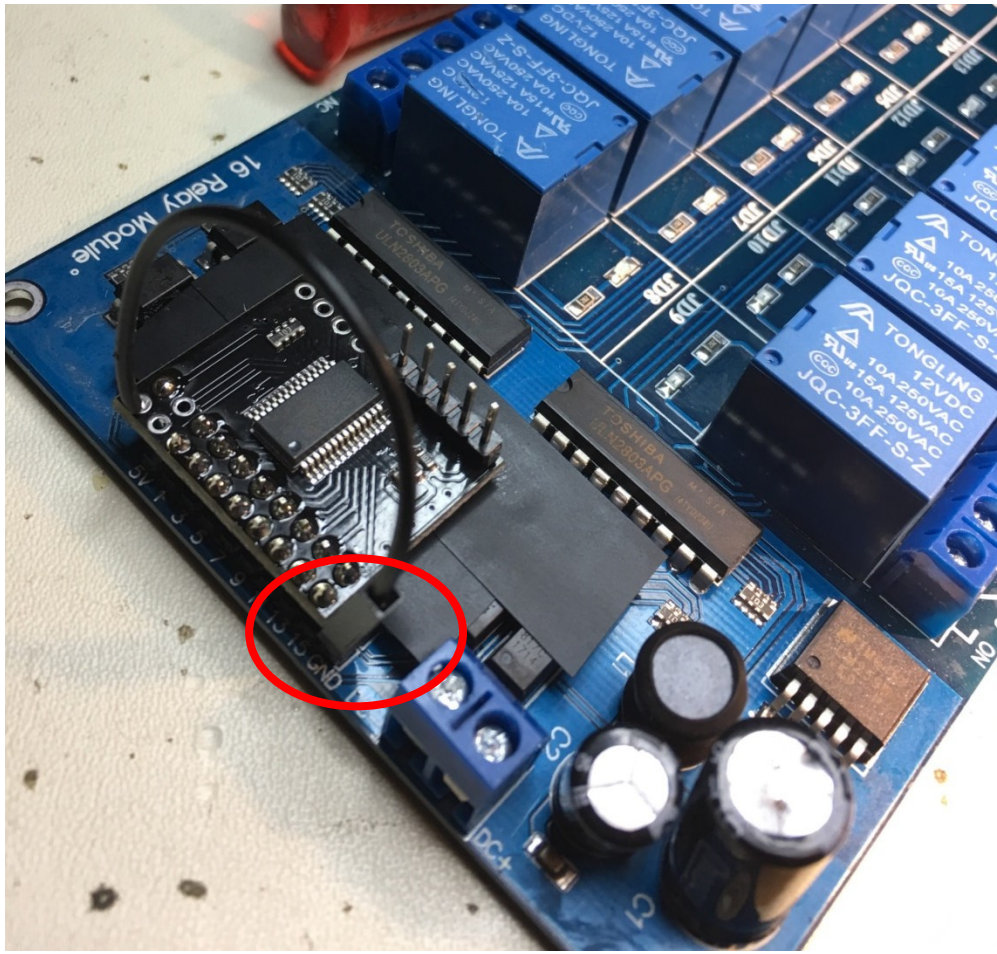






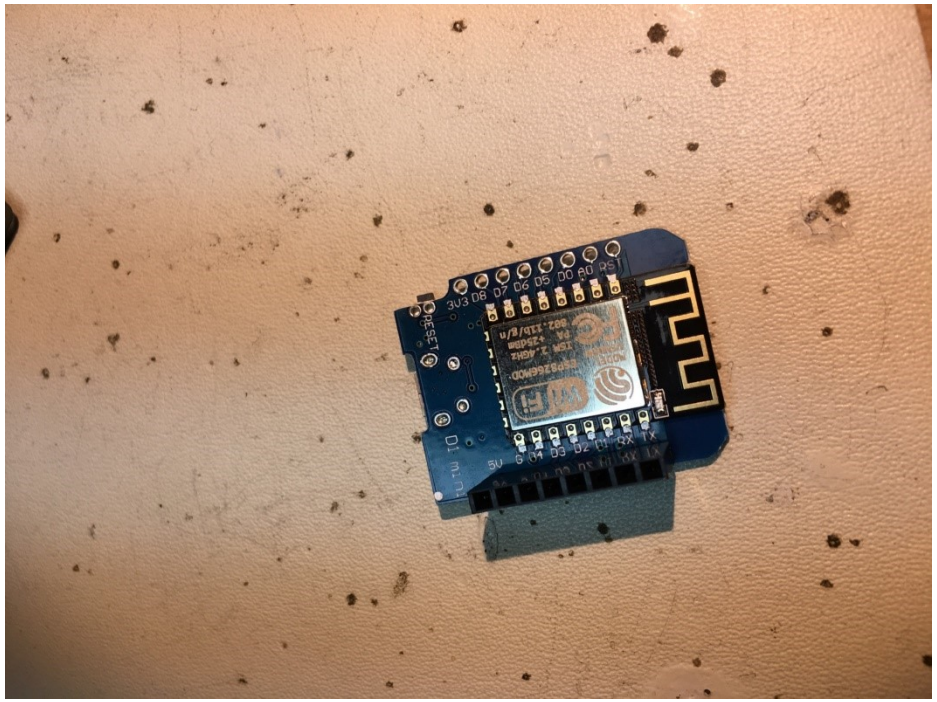
Soldering MCP-23017 board onto your RelayBoard

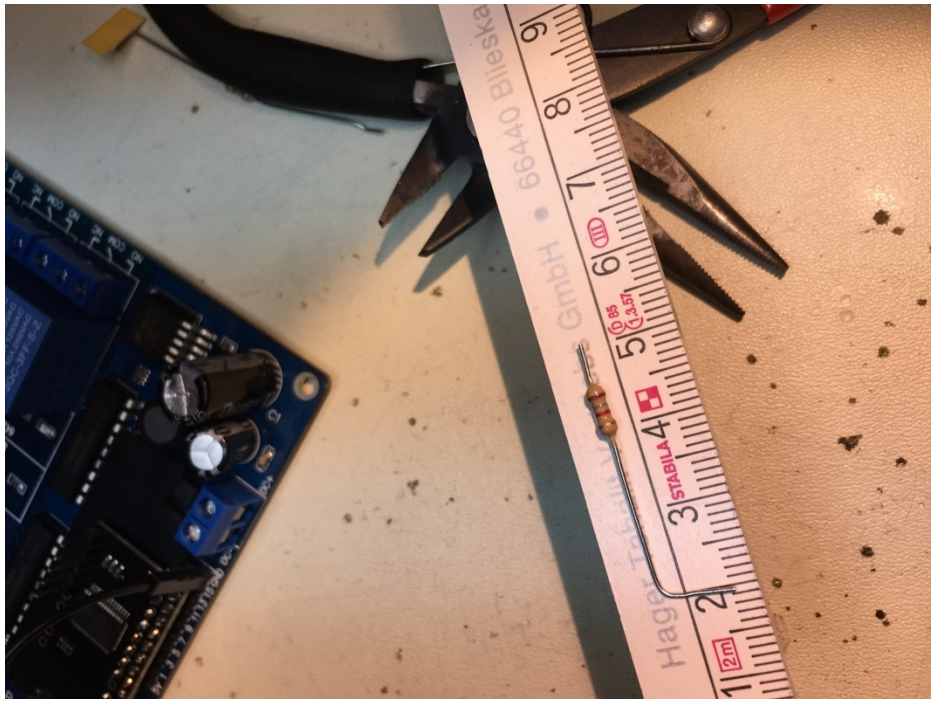
- Use tape to reduce the risc of shortcuts
- Place your MCP-Board correct
- Solder all MCP-PCB connections to your Relayboard
- Connect GND-Wire with your RelayBoard



Preparing your ESP-12F… 1

- Solder Female Header & prepare your 1k8/2k2 resistors:



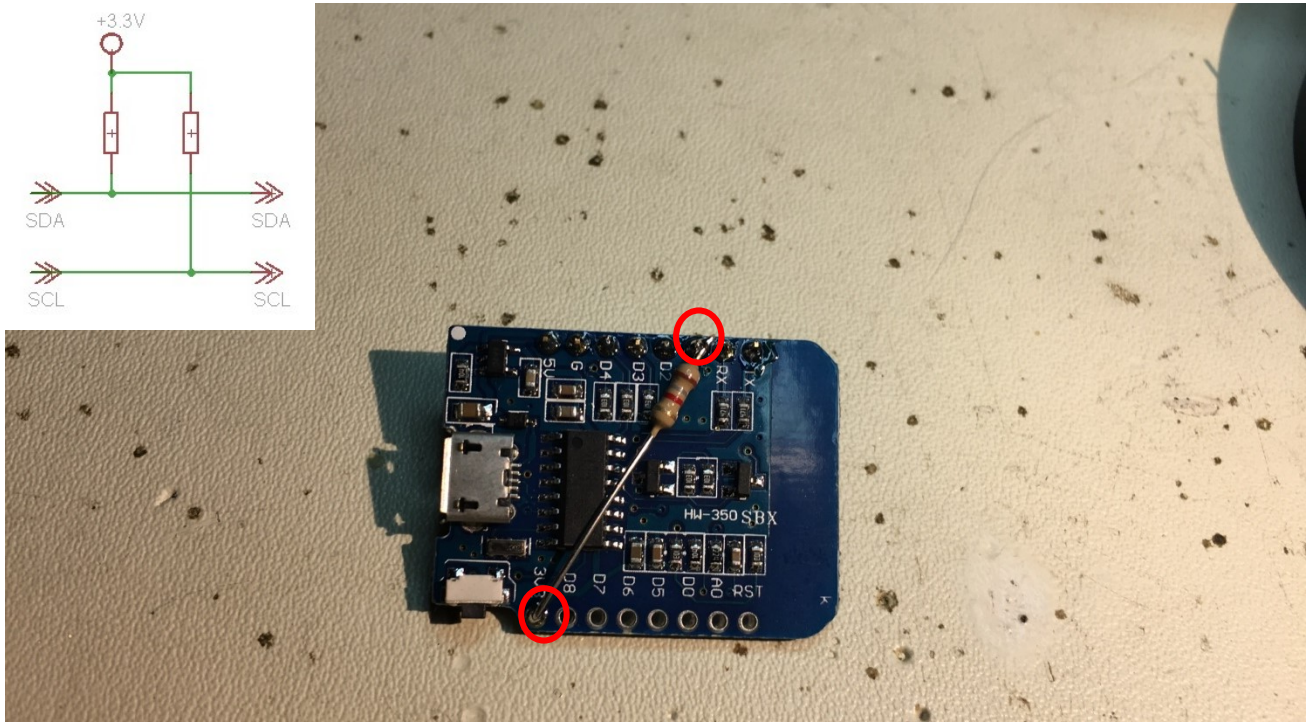


Preparing your ESP-12F… 2

- Solder your first 1k8/2k2 resistor.

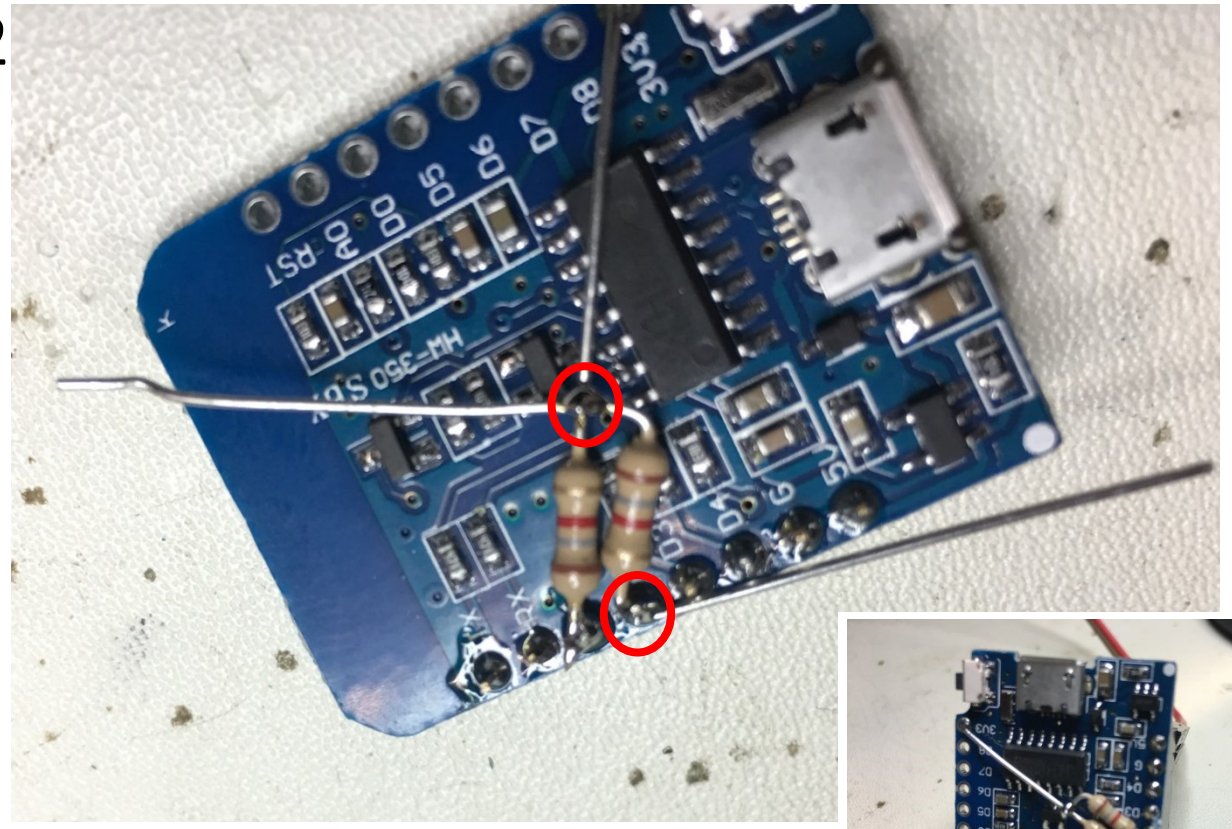
  Pin D1 ➔ 3V3!

- Be careful :P…



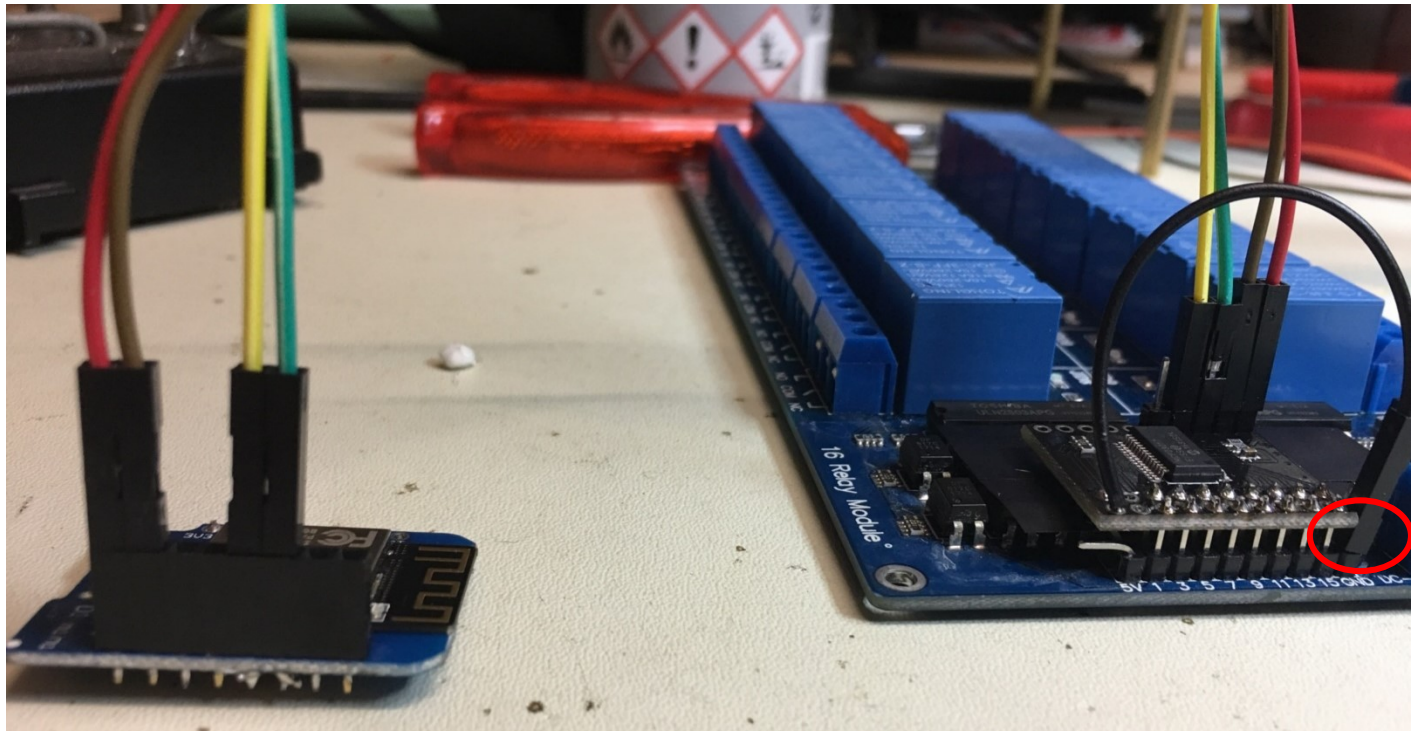
Preparing your ESP-12F… 3

- Bent your second resistor 1k8/2k2 into an „S“ and solder it.
- D2 ➔ 3V3
- Cut.
- Be careful :P…



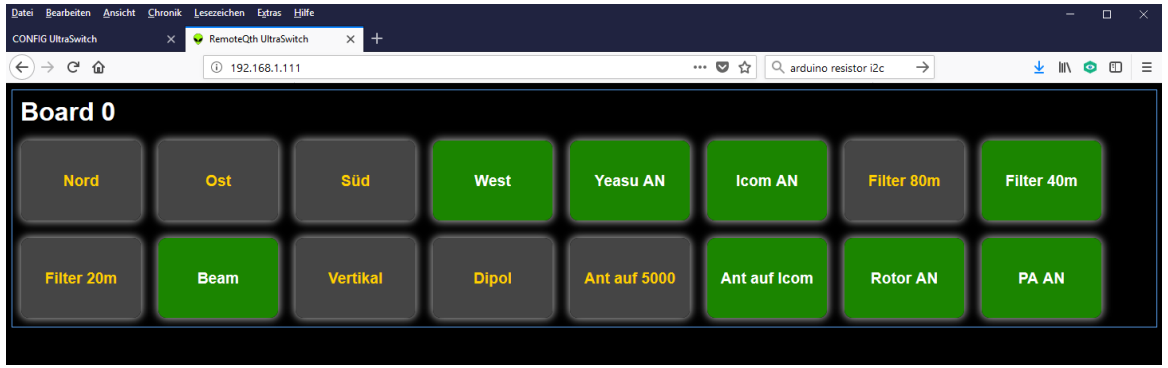
Put it all together…

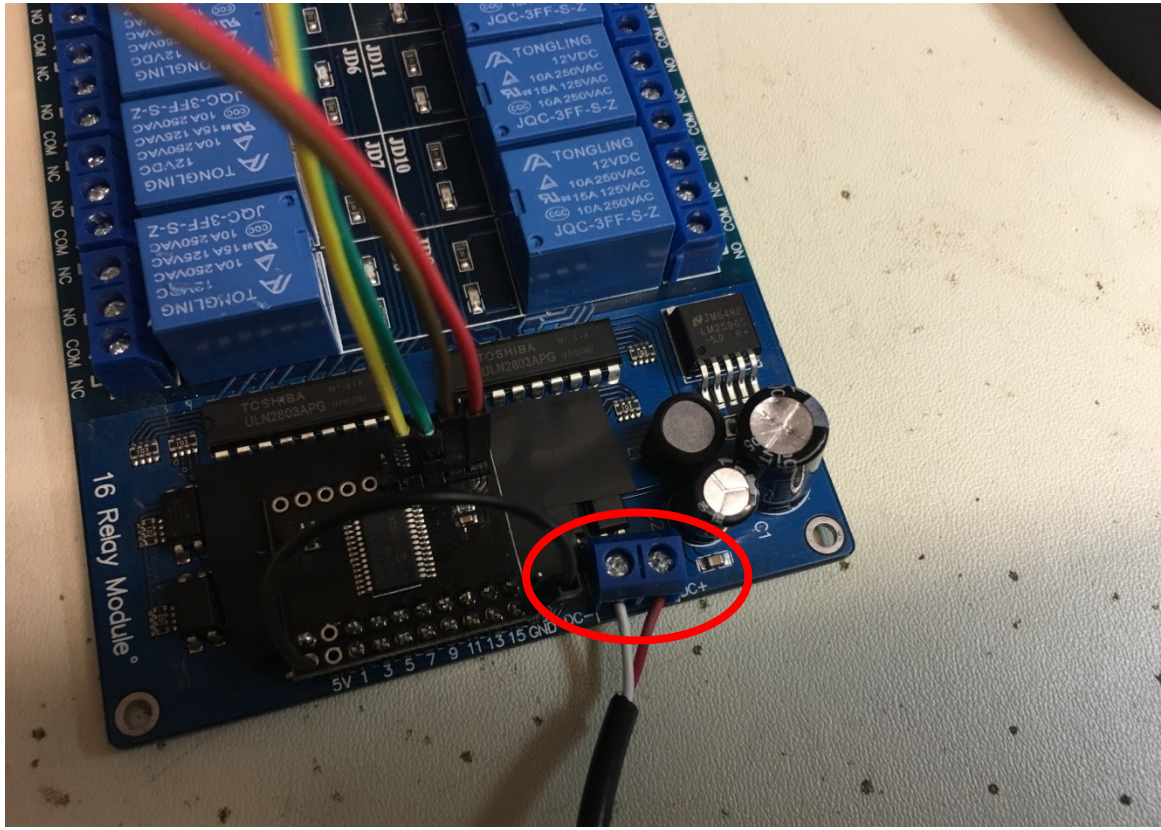
- Wemos from left to right (example):
  5V|GND|empty|empty|yellow|green
- MCP from the left:
  Yellow|Green|GND|5V
- SCHWARZ to GND (Relayboard)
- Your wire colors may be different!



Finally…

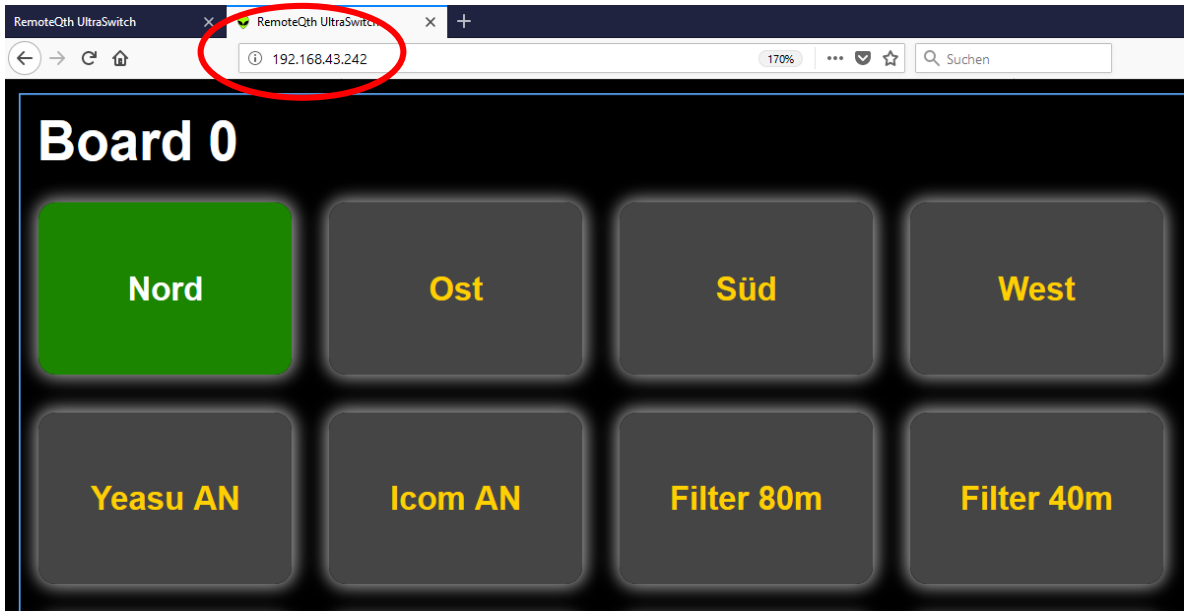
- Connect your power supply 12V
- Power on
- Start your browser
- Be happy! ☺





Back to the beginning… Use Case 1

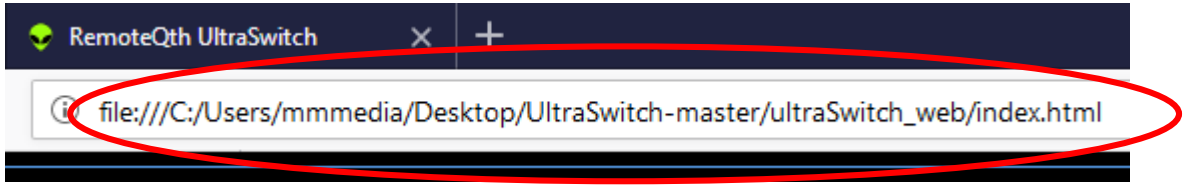
- Call your ultraswitch by its ip
- The Ultraswitch returns his index.htm
- Labels, groups, … are loaded from h.mmmedia-online.de



⇒No customizing necessary if this fulfills your needs..

⇒This should be step 1 befor customizing!

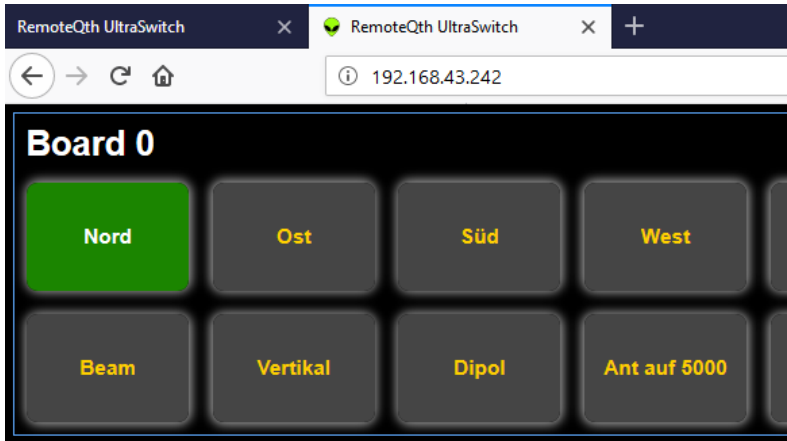
Back to the beginning… UseCase 2… 1

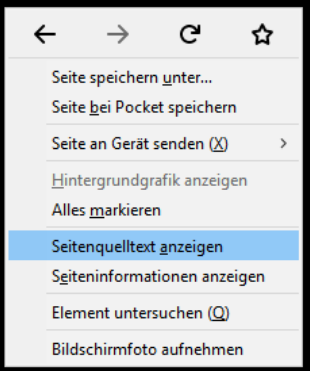


- The Ultraswitch is called using your local stored (eg. desktop) index.html
- E.g file:///C:/Users/mmmedia/Desktop/UltraSwitch-master/ultraSwitch_web/index.html
- All files can be local: Customize!

Back to the beginning… Creating your own index.html… 1

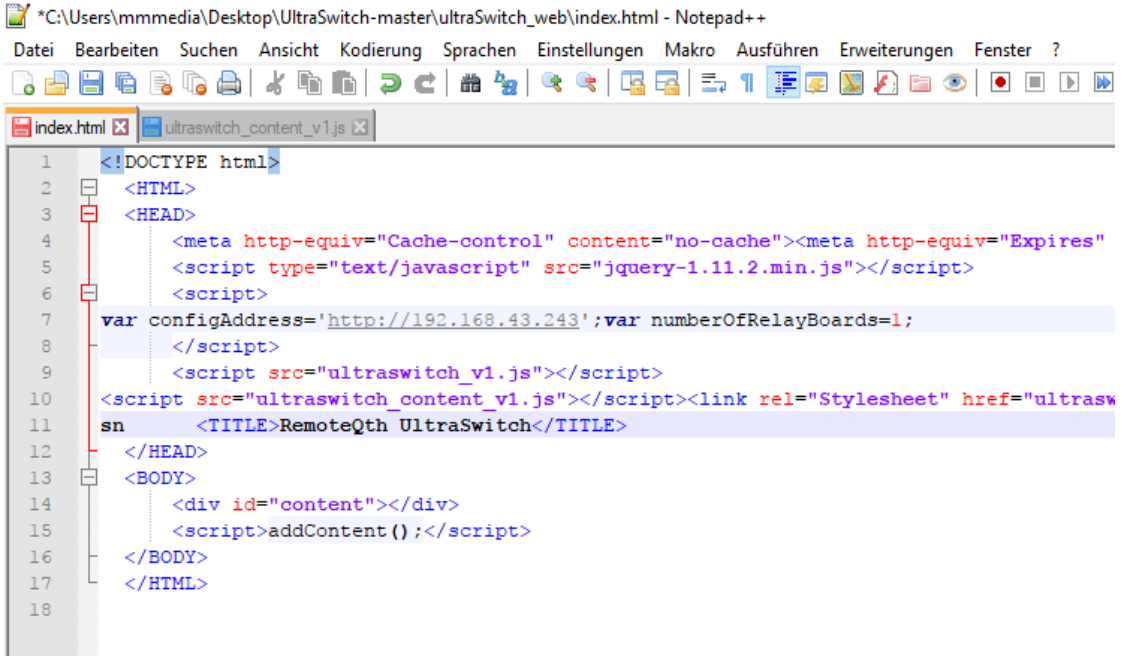
- Call Ultraswitch by IP e.g. 192.168.43.242
- Right-Click somewhere to the black backround of the webpage and say „show source code“
- Copy the complete text into your clipboard(ctrl+a, ctrl+c)





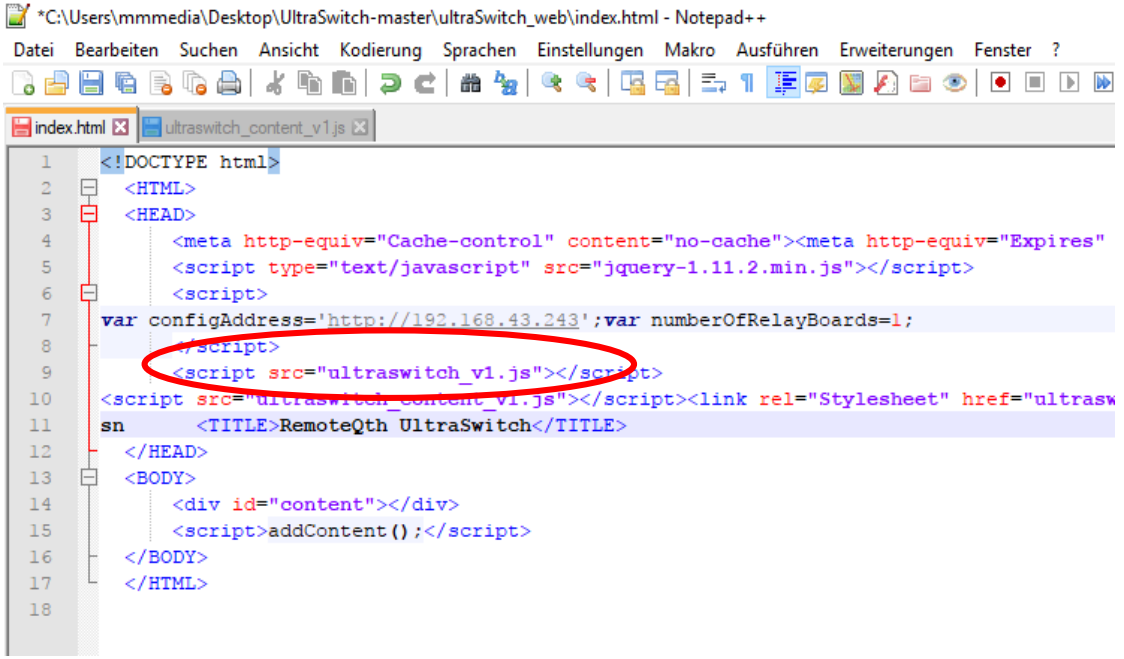
Back to the beginning… Creating your own index.html… 2

- Open th index.hml from your ultraswitch_web folder with an Editor .
- Remove all the code
- Past the text from your clipboard (ctrl+v)
- Open the new index.html. All should be visible like before – if not look into CORS (see slide about CORS)



UseCase 2... All local!

- If all needed files are all in the same local directory like your index.html, than you can change the file paths like this:
  e.g. change *src=„http://h.mmmedia-online.de/ultraswitch_v1.js“* to this *src=„ultraswitch_v1.js“*
- If all http://<url>/<filename> changed into <filename>, all is working without the internet
- Problem: jquery must be local, too! To be completly independent from the internet.
- Ask someone you who knows how to do web pages for help.



Customizing => Labels, Groups, RelaysPerBoard

Customizing your **ultraswitch_content_v1.js**:

- Labels => textboard0 = Labels for the 1. Board etc…
  var textboard0 = ["Nord","Ost","Süd","West","Yeasu AN","Icom AN","Filter 80m","Filter 40m","Filter 20m","Beam","Vertikal","Dipol","Ant auf 5000","Ant auf Icom","Rotor AN","PA AN"]; ➔ Change it, but keep it short!!

- Groups => board0Group = Grouping pins definition for your 1. Board…
  var board0Group = [[0,1,2,3],[6,7,8],[9,10,11],[12,13]];

  What does it mean: On Board 1 with the id 0 are 4 groups defined. First group: button 0,1,2,3 is connected together in a way “there can be only one”-

- numberOfRelaysPerBoard = How many Relays per Board (1., 2., 3.,…)
  var numberOfRelaysPerBoard = [14,16,16,16,16,16,16,16,16,16,16,16,16,16,16,16];

  What does it mean: Your first board controls 14 Relays, 2nd 16, ….

UseCase 3... Using your own Webspace /webserver

Your own Webserver (e.g. RaspiPI):

- Use Config.html to point the content IP-Adresse to your OWN WEBSERVER.
- **The URL to ultraswitch_content_v1.js** and **ultraswitch_v1.css** is generated by naming convention: http://<own_ip>/h/<filename>!

HowTo:

- Create a subdirectory named „h“ in your webserver/webspace root.
- Copy your own **Ultraswitch_content_v1.js** und **ultraswitch_v1.css** into this folder.
- **Call Ultraswitch just with its IP! Its not needed to call the Ultraswitch by your local index.html (but you sure can)**
- If you are using a Raspberry PI , hosting all files there makes sense (incl. your own index.html)

Trouble with the Ethernet-Lib…

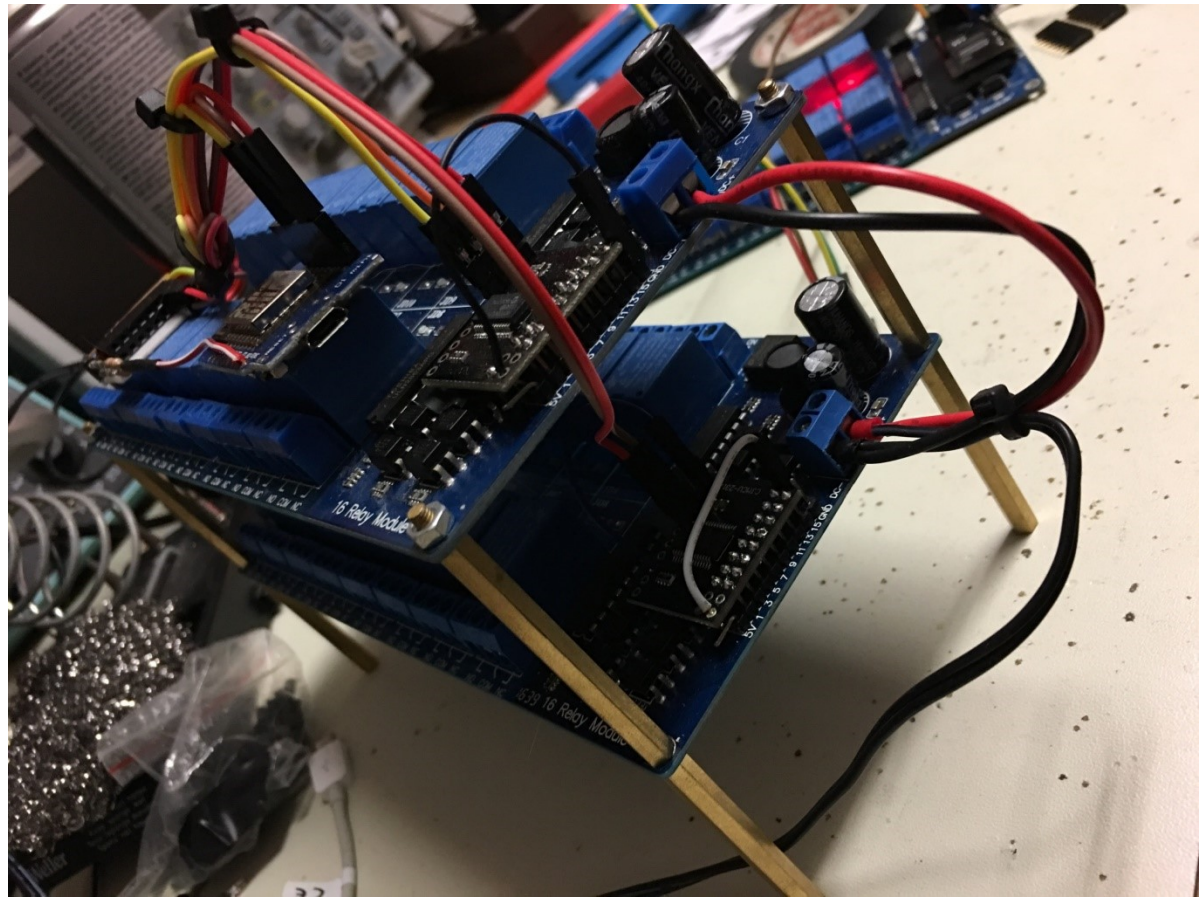
- Due to the age of the Ethernet-Lib implementation, the lib has some problems with dealing with modern browsers – especially with the number of connections opened.
- The Ethernet lib can handle 4-10 connections, depending on the ethernet chip. But a modern browser opens dozens over dozens.
- Too many open connections create a problem with the number of open sockets. Race conditions ocour – the System is not answering after a while.
- Workaround: MicroFox (see folder „Software“) – a portable customizes version of the firefox. The number of max-connections is set to 2. This is very good calling your ultraswitch, but not very cool while surfing. So just use this browser for communication with the ultraswitch only if you are running into connection problems. (Your mobile firefox can be customizes too – see about:config and search for max-connections)

Architecture of the source code

- Main-Sketch is used to control the ESP12F hardware, JSON, WebServer:
  UltraSwitch.ino
- Liraries:
  Adafruit-MCP23017 ➔ I2C control MCP23017
  ArduinoJson ➔ JSON for the Webserver-Response
  WiFiManager ➔ Easy setup your wifi webserver
- Ultraswitch_content_v1.js ➔ Configure Labels, groups, etc. (*Naming*)
  Ultraswitch_v1.js ➔ Switching logic and controlling UI-Dom-Elements
  Ultraswitch_v1.css ➔ UI Look&Feel, buttonsize, colours, etc. (*Buttons*)
- Setup_esp.html ➔ Preconfigure your ESP12F Access-Points
  Config.html ➔ Configure IP- and Serverpreferences
  index.html ➔ Local ESP12F application

Extention with more 16CH modules…

- Build more Relayboard+MCP units
- Set their unique I2C-Adress! (see next slide)
- Just connect the 2 I2C-cables for SCA and SCL (Parallel-Bus)
- Attach 12V
- Configure Config.html ➔ Set the amount of boards
- done! (Without a firmware upload)

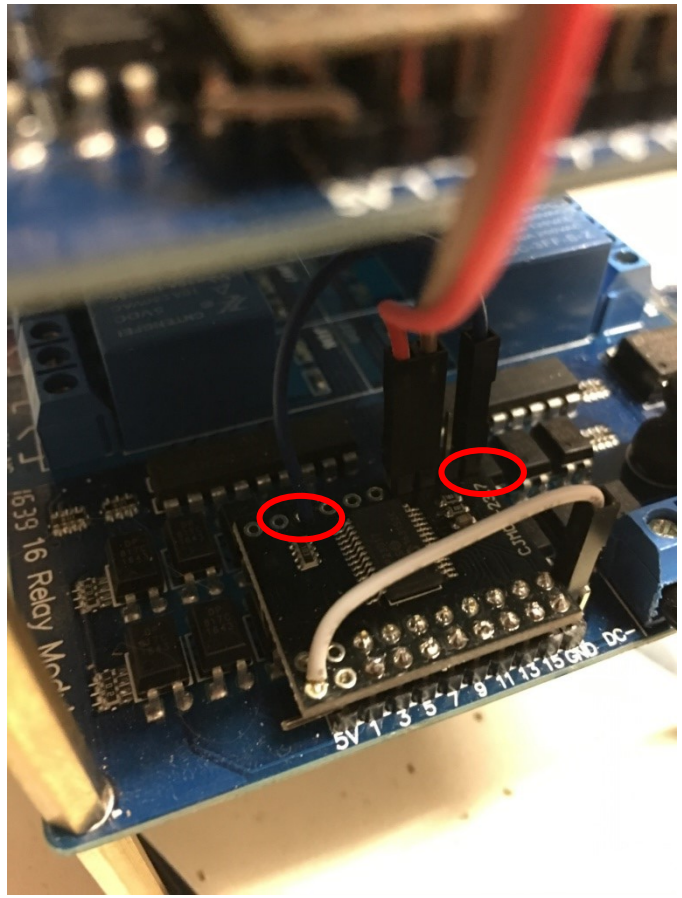


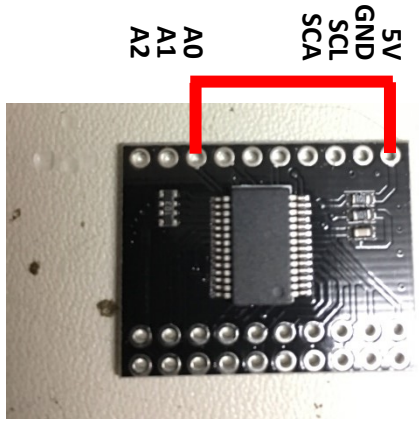
More moduls… Set the I2C adresses => Up to 8 devices!

- With setting the desired Pin to HIGH (5V) you can control how many boards are connected:
  0 => 1st board, 1 => 2nd board,
  2=> 3rd board, …!!!

```
addr 0 = A2 low , A1 low , A0 low  000
addr 1 = A2 low , A1 low , A0 high 001
addr 2 = A2 low , A1 high , A0 low  010
addr 3 = A2 low , A1 high , A0 high  011
addr 4 = A2 high , A1 low , A0 low  100
addr 5 = A2 high , A1 low , A0 high  101
addr 6 = A2 high , A1 high , A0 low  110
addr 7 = A2 high, A1 high, A0 high 111
```

Hint: Not connected => LOW





A lot of space to create your own projects…

- A Ultraswitch-Cluster
- Using the ADC-IN port for measurements (Temp, etc.)
- For antenna switching (Hint: Shielding, Housing…)
- Time controlled Pins (see mega-webswitch)
- Integration into N1MM (ask M. Hammelmann, DK5AX)
- Remote-Switch for Remote Stations (Using VPN is highly recommended)
- Integration with a Raspberry PI as your server (UDP, Broadcast)
- …

THE END

- If you are running into a problem, contact us. We will never leave you alone! –

More projects at: www.remoteqth.com
Want to buy a kit? More questions? Email: dm5xx@gmx.de -